8-1-1991

Substitution of the statistical range for the variance in two local noise smoothing algorithms

Marc R. Lapointe

SUBSTITUTION OF THE STATISTICAL RANGE FOR THE VARIANCE IN TWO LOCAL NOISE SMOOTHING ALGORITHMS

by

Marc R. Lapointe

B.Eng. Royal Military College
of Canada

(1980)

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science in
the Center for Imaging Science in the
College of Graphic Arts and Photography of
the Rochester Institute of Technology

August, 1991

Signature of the Author _____
Center for Imaging Science

Accepted by Mendi Vaez-Pravani
Coordinator, M.S. Degree Program

COLLEGE OF GRAPHIC ARTS AND PHOTOGRAPHY
ROCHESTER INSTITUTE OF TECHNOLOGY
ROCHESTER, NEW YORK

CERTIFICATE OF APPROVAL

M.S. DEGREE THESIS

The M.S. Degree Thesis of Marc R. Lapointe
has been examined and approved
by the thesis committee as satisfactory
for the thesis requirement for the
Master of Science degree

Dr. Edward M. Granger, Thesis Advisor

Dr. John R. Schott

Mr. Peter G. Engeldrum

14 Aug 91
Date

THESIS RELEASE PERMISSION FORM

ROCHESTER INSTITUTE OF TECHNOLOGY

COLLEGE OF GRAPHIC ARTS AND PHOTOGRAPHY

Title of Thesis Substitution of the statistical range for the variance in two local noise smoothing algorithms

I, Marc R. Lapointe, hereby grant permission to the Wallace Memorial Library of R.I.T. to reproduce my thesis in whole or in part. Any reproduction will not be for commercial use or profit.

Date 14 Aug 91

SUBSTITUTION OF THE STATISTICAL RANGE
FOR THE VARIANCE IN TWO
LOCAL NOISE SMOOTHING ALGORITHMS

by

Marc R. Lapointe

Submitted to the
Center for Imaging Science
in partial fulfillment of the requirements
for the Master of Science degree
at the Rochester Institute of Technology

ABSTRACT

The statistical range was substituted for the variance in local noise smoothing algorithms proposed by Tomita and Tsuji, and by Nagao and Matsuyama in order to reduce computer processing time. Images of low, medium, and high information content were corrupted by two levels of combined additive and multiplicative noise and were processed by the original and modified algorithms for both the uncorrelated and correlated noise cases. A subjective paired comparison was performed on the resulting images based on four evaluation criteria, and a normalization of the data between observers as well as hypothesis testing were carried out. Under the conditions of the test, it was found that comparable noise smoothing performance could be achieved only with the algorithm by Tomita and Tsuji, but that noted improvements occurred in the areas of preservation of subtle details, immunity to shape distortion and preservation of step edges.

ACKNOWLEDGEMENTS

Successful completion of this thesis owes recognition to the support from many sources. Valuable assistance and advice was particularly appreciated from the following:

Dr. Edward Granger for graciously accepting to act as thesis advisor and for his direction in the conduct of the analysis of results;

Dr. John Schott and Peter Engeldrum for their interest and most useful discussions and suggestions;

Nitin Sampat for supplying the original digitized images and computer subroutines used during this thesis work;

John Francis for his invaluable assistance in the handling of digitized images on RIT's Gould DeAnza image processing systems;

the following observers for their gracious and patient participation in the subjective comparison of images: Denis Daoust, Wayne Farrell, John Francis, Patrick Friedman, Tim Hawes, Gordon Leggett, Ricardo Motta, Nimi Natan, Edward Pariser, Wendy Rosenblum, Carl Salvagio, Dr. John Schott, Greg Snyder, John Thayer, and Juan Zuleta.

The support of the Canadian Forces Educational Grant Program (contract CF-8402-PG-41.G1) is also acknowledged with appreciation.

TABLE OF CONTENTS

LIST OF APPENDICES

LIST OF FIGURES

LIST OF TABLES

I. INTRODUCTION

1. GENERAL

Continuous-tone imagery can be represented digitally by two-dimensional arrays of numbers, each number representing the average gray level of the image over the area of a picture element (or 'pixel') at a given location in the image. In its broadest sense, digital image processing is then concerned with the manipulation of these arrays of numbers by means of a digital computer or other appropriate hardware processor; and one purpose of digital image processing is to improve picture quality by image restoration and image enhancement techniques.

A. Image Restoration

Image restoration is concerned with the ability to restore an image to its original quality according to mathematical manipulations intended to invert physical degradation phenomena experienced in the formation of the image in the first place.[1] The types of degradation encountered in image formation may be linear (defocus, linear motion, spherical aberration, atmospheric degradation) and non-linear (geometric distortion, non-uniform or turbulent atmosphere, non-linear motion degradation, coma, tilt in cylindrical lenses).[1] The emphasis is then on degradation

modeling and on recovery of the true image by inversion of the degrading process.[3]

B. Image Enhancement

On the other hand, image enhancement in general extends the human interpretation ability and increases the chance of success in automatic picture processing.[5] It is designed to enhance or in some way alter the quality and features of an image on the basis of the psychophysical characteristics of the human visual system, without recourse to the knowledge of a degradation process.[1,2,4] The improvement in the quality of the image is often subjective and is related to the application as well as the judgement of the viewer. For instance, the sharpening of edges in an image that appears blurred may be required in one application, whereas another application may defocus the image so that sharp details and noise are eliminated, making other features more detectable.

Image enhancement techniques may be organized in four groups.[5] They are:

a. Spatial smoothing of regions, which employs linear or non-linear spatial-domain low-pass filters;
b. Gray-level rescaling, which manipulates or requantizes gray levels for contrast enhancement;

c. Edge enhancement, which involves linear or non-linear spatial-domain high-pass filters;

d. Frequency-domain filtering, which utilizes low or high-pass filters in the frequency domain where Fourier transformation is required.

This study will deal with spatial smoothing of regions. A brief review of current techniques will be given as well as an overview of noise and statistical considerations relevant to the study.

2. NOISE CONSIDERATIONS

Noise may be generated from various sources including the particulate nature of the film, photon and other fluctuations in scanning light, and electrical sensor noise.[2,7]

Some types of noise encountered in digital images are additive (independent of the picture signal), but others are multiplicative. Some are uncorrelated from pixel to pixel, while some are correlated (or "coherent").[8]

A. Additive Noise

Channel noise and photodetector noise are generally

independent of the picture signal and the resulting gray levels at each pixel can be represented by [24]

$$z_{ij} = x_{ij} + w_{ij} \tag{1}$$

where x_{ij} is the input signal

w_{ij} is random additive noise

and where the expected value (or the mean) of the noise is zero:

$$E(w_{ij}) = 0 \tag{2}$$

where $E(w_{ij})$ is the expected value at pixel location i,j.

For simplicity, most authors assume that the noise is random, uncorrelated, additive, and normally distributed with zero mean.[5,6]

B. Multiplicative Noise

A common example of multiplicative noise is photographic granularity, where images are formed by randomly shaped and sized clumps of developed silver grains in the emulsion. In order to describe the granularity phenomenon, random dot models have been suggested for various imaging systems.[26] The exact random dot model assumes opaque dots and predicts that the granularity increases exponentially with the average

density of the film. On the other hand, a random dot model which assumes partially transmitting dots predicts a decrease in granularity as the density increases past a certain point. The combined effects of these two models offset each other, and experimental results will usually follow the Siedentopf relationship where the granularity is a linear function of the density.[27] Curves representing the relationship between the granularity (G) and the density for the Siedentopf relationship and the exact random dot model are shown in figure 1.

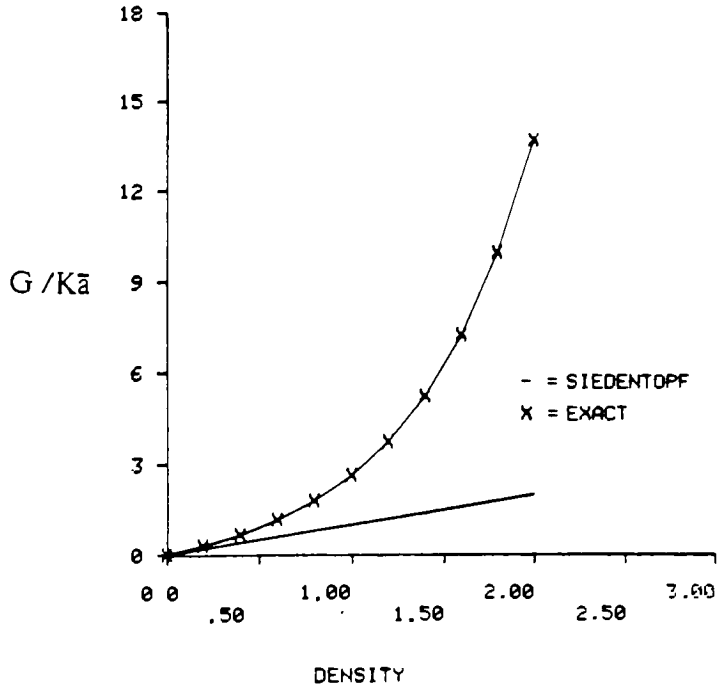


Figure 1. Normalized Granularity vs. Density curves for Siedentopf and Exact Random Dot Model (from Engeldrum, reference 26)

Mathematically, images with multiplicative type noise have been represented in the literature by [8,24]

$$z_{ij} = x_{ij} + v_{ij}\, x_{ij} \quad (3)$$

$$= x_{ij} \,(1 + v_{ij}) \quad (4)$$

$$= x_{ij}\, u_{ij} \quad (5)$$

where x_{ij} is the input signal

v_{ij} is a multiplicative factor (function of x_{ij})

$u_{ij} = 1 + v_{ij}$

Following this approach, Lee[24] used a linear approximation to represent multiplicative noise, with experimental results reported as "promising".

Upon examination, equations (3) to (5) only include a multiplicative term, and represent a re-mapping of the gray levels without introducing a random element, or noise. Images with a combination of multiplicative and random additive noise can be represented mathematically by:

$$z_{ij} = x_{ij} + k(x_{ij})\, w_{ij} \quad (6)$$

where x_{ij} is the input signal

w_{ij} is random additive noise

$k(x_{ij})$ is a multiplicative factor (function of x_{ij})

Note that in the last equation, the multiplicative factor is still a function of the input signal but is applied to the random noise instead of the input signal. It is assumed in this study that the gray levels x_{ij} and z_{ij} represent density in the digitized images (as opposed to transmittance), and that we are in the linear portion of the D-log H curve.

C. Correlated Noise

Uncorrelated noise is considered by most authors for simplicity. However real-world images may exhibit varying degrees of noise correlation from point to point. Correlated noise will occur when an uncorrelated noise distribution is convolved with a spread function of finite dimensions.[28] This is the case when an image is scanned with an aperture of finite size during the digitization process, corresponding to some low-pass filtering of the uncorrelated noise. Electronic noise on a signal may also be highly correlated (eg. some video noise).

3. SPATIAL SMOOTHING OF REGIONS

Spatial smoothing of regions is used to remove noise from an image. Generally, digital image smoothing techniques fall into two categories.[6] In the first one, the noisy image is

processed globally in the sense that the whole or a large portion of the image is correlated to produce a smoothed image. This is the case when Weiner (or Least Squares) or Kalman filters are used. In the second category local operators are applied to the image, where only those pixels in a small neighborhood of the concerned pixel are involved in the computation.

The simplest smoothing technique is equal-weighted averaging over a neighborhood, where the gray level of the pixel of interest is replaced by the average of those in a rectangular neighborhood surrounding it.[5] This technique will remove noise effectively, but will also blur the image, especially at the edges of objects.

To reduce the blurring effect, unequal-weighted smoothing techniques can be used, where the gray level of the pixel of interest is weighted more than those of its neighbors. Context-free techniques have been proposed by Graham[9] and Brown[10], using a 3x3 neighborhood and factor matrices with constant coefficients. They will not remove noise as efficiently as the equal-weighted averaging method, and they blur the edges although not to the same extent. Lev et al[11] proposed a 3x3 context-sensitive weighting-matrix where the weighting is reduced in the direction which shows a sharp change in gray level (e.g. over an edge). The blurring effect

at edges can be reduced, but the technique is computationally complicated and requires the specification of a prescribed parameter. Gradient Inverse Weighted Smoothing, a computationally simpler context-dependant 3x3 weighting-matrix, was proposed by Wang et al.[12] It smoothes images with little blurring and does not require prespecified parameters.

Thresholding has been suggested to clean up "salt and pepper" noise.[7,9,14] Pixels having a gray level much lower or higher than the average of their neighbors are assigned the average gray level. This technique requires the specification of a threshold value and prior knowledge of image characteristics (context-dependant), and will also remove small details above the threshold value thereby reducing the resolving ability in higher contrast areas.

Weighted smoothing procedures based on subjective criteria were suggested by Anderson and Metravali,[18] and by Trussell,[19] based on the eye tolerance to noise in areas of high signal strength. The technique is reported suitable for many images, but is less effective if the noise is not distributed uniformly over the image.

Median filtering, a non-linear technique introduced by Tukey,[20] assigns the median gray level of a neighborhood to the

central pixel. With this technique however, if a detail has a width of less than one-half of the window width, it is simply suppressed. To avoid this signal suppression, variable window widths were proposed by Pratt,[7] but at the expense of computation time.

Graham approximated the second-order derivatives of pixel gray level in the x and y directions by defining measures of "flexure" delta-x and delta-y.[5] Equal-weighted averaging is then performed over a specific portion of the neighborhood (e.g. over a horizontal or vertical line) based on a decision test that compared the flexure with a prespecified threshold parameter. This method only smoothes homogeneous areas (i.e. with small second derivatives), ignores isolated noise pixels that are much darker or brighter than their neighbors and depends on the specification of a thresholding parameter, requiring a priori knowledge or trials.

A scheme was proposed by Kuwahara et al,[15] and by Tomita and Tsuji[16] where the gray level of the pixel of interest is replaced by the average gray level of its most homogeneous neighboring region. Kuwahara et al used a four region neighborhood, while Tomita and Tsuji added a fifth central region, as depicted in figure 2 below. For each of the 3x3 regions in the neighborhood, the mean gray level and the variance are calculated and the central pixel is assigned the

average gray level of the region with the lowest variance, assuming it is the one containing no sharp boundaries. The technique removes noise and also preserves the image's edges, provided it is not applied to images with complex-shaped boundaries, where large rectangular regions are not very effective.

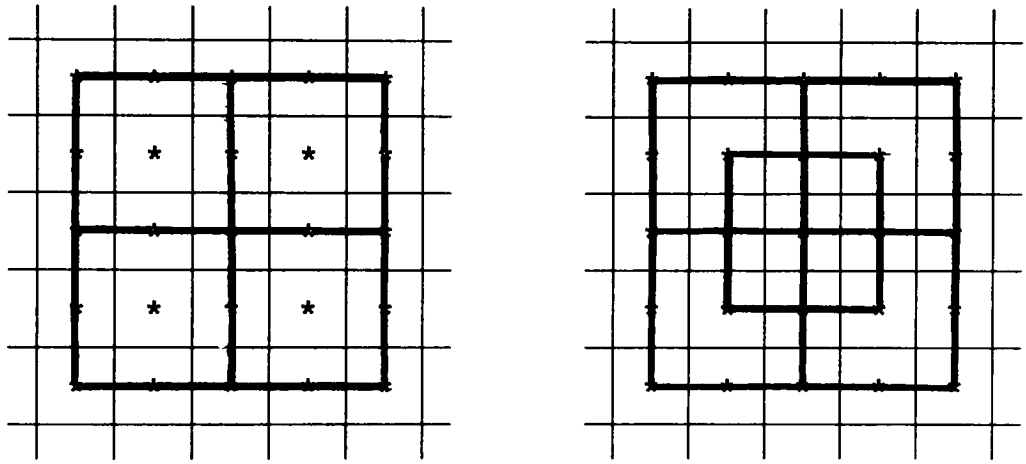


Figure 2. Neighboring regions used by:
(a) Kuwahara et al, and
(b) by Tomita and Tsuji.

To improve on the above two methods Nagao and Matsuyama[17] proposed a more directional filter by dividing a 5x5 neighborhood into nine regions (four pentagonal, four hexagonal, and one 3x3 square region) as depicted in figure 3. Eight of these areas in fact simulate the rotation of an elongated bar mask around the pixel of interest (see figure 4), which will smooth a region without blurring sharp edges nor destroying the shape of a complex boundary.

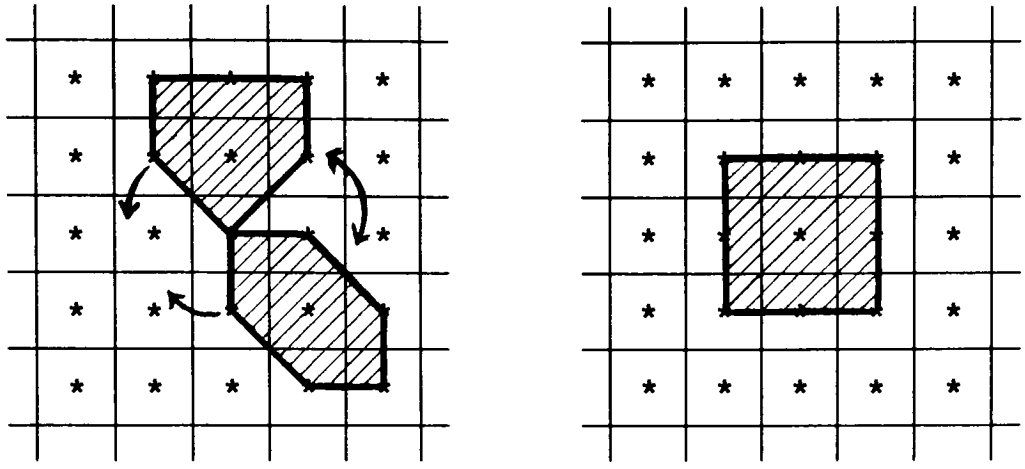


Figure 3. Regions used in the Nagao and Matsuyama technique.

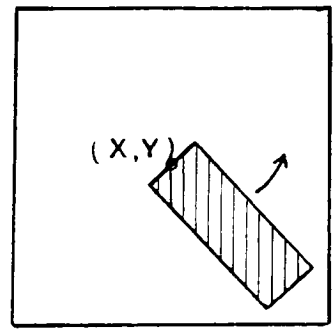


Figure 4. Elongated bar mask representation.

Again, the central pixel is assigned the average gray level of the most homogeneous region, also determined by calculation of the variance. A 5x5 window is used around the pixel of interest to preserve small details and complex edges, however noise cannot be strongly reduced with a single pass due to the small size of the region over which the averaging is done. Nevertheless, repeated iteration of this smoothing

operation is reported to produce the same effect as averaging once over a larger neighborhood. The technique is therefore repeated until the gray levels of almost all pixels do not change. This method is reported as yielding better results than the previous two,[17] but is computationally complex and time consuming.

4. HYPOTHESIS

In this study, the algorithms proposed by Tomita and Tsuji, and by Nagao and Matsuyama, will be modified as follows: the modified algorithms will use the statistical range (difference between the highest and the lowest pixel value) instead of the variance to determine which neighborhood surrounding the pixel of interest is the most homogeneous. The neighborhood with the lowest range will be assumed to be the most homogeneous and the average gray level of all its pixels will be assigned to the pixel of interest. It is suggested that the above substitution in these two algorithms will yield images of comparable quality. This hypothesis is based on the following statistical notions.

When sampling discrete variables the population standard deviation σ may be estimated from the sample standard deviation s defined by the formula[21]

$$s=\sqrt{\frac{\sum_{i=1}^{n}(x_i-\bar{x})^2}{(n-1)}} \qquad (7)$$

where x_i is the value of one element of the sample

$\bar{x}$ is the mean of the sample

n is the number of elements in the sample

and the sample variance is the square of the sample standard deviation.

Besides s, the sample range R (difference between the highest and lowest values in the sample) may be used as an estimator of the population standard deviation. Given a random sample of size n from a normal population, it can be shown that the statistic R/d_2 is an unbiased estimator for σ, where d_2 is a constant which depends on the size of the sample n as shown in the following table.[22]

TABLE 1 : d_2 values as a function of sample size n

n	2	3	4	5	6	7	8	9
d_2	1.128	1.693	2.059	2.326	2.534	2.704	2.847	2.970

For very small samples ($n<6$), R/d_2 is nearly as good an estimator of σ as s.[22]

In smoothing schemes proposed by Tomita and Tsuji[16] and by Nagao and Matsuyama,[17] the calculation of the variance is required for each of the five and nine regions of the neighborhood respectively, to find which one is the most homogeneous. This is computationally costly, and these methods would be more attractive to users should a faster scheme be provided. The number of sample points is nine in all regions of Tomita and Tsuji's algorithm and seven and nine points for Nagao and Matsuyama's algorithm. Since a relatively small number of points is involved in each region and given the assumption that the distribution of the gray levels is normal in a homogeneous region of a digital image, then it is suggested that the use of the R/d_2 statistic may be used as a satisfactory decision criterion in the determination of the most homogeneous region in the above two techniques if it were to be used in lieu of the variance. This would thereby reduce computation requirements and still yield acceptable noise smoothing capability when compared to the original algorithms.

The following sections will deal with the implementation of filtering algorithms on noise corrupted images, and the objective and subjective evaluation of the resulting images in

order to verify the above hypothesis.

II. METHODS

1. SELECTION OF DIGITAL IMAGES

Three different images were used in this study to represent various levels of information content (i.e. spatial resolution). The first one was a gray scale, the second one was that of a girl's face and the third one was a picture of a city center scene, representing low, medium and high information content images respectively. The filtered versions of all three images will be used in a subjective evaluation, as described later. The first image will also be used for an objective evaluation. Appendix 1 shows the images used, prior to noise corruption.

These images had already been digitized and were provided to the author from computer memory by Nitin Sampat, Graduate Student of the Imaging and Photographic Science Program, who had obtained them from his thesis advisor. The images were stored in the author's computer account on RIT's computer system.

The digital images were 512 by 512 pixels in size, and had an eight-bit radiometric resolution (256 gray levels) in order to achieve continuous tone representation when displayed on a video monitor or made into photographic prints. They

were stored as eight-bit binary type (LOGICAL*1) data to keep storage requirements to a minimum.

The storage capacity required on a VAX computer system for one eight-bit, 512 by 512 pixel image is 512 user blocks. The storage capacity on the author's computer account was set at 80,000 blocks on VAX A, B, and C to store the various images generated during this study, and to 15,000 blocks on VAX D to store the computer programs.

2. CREATION OF NOISE FILES

A. Uncorrelated Noise

Digital white noise with a Gaussian distribution of gray levels was artificially generated by using the random number generator, a built-in function of the VAX computer. The random generator provides a uniform distribution (i.e. equal probability of occurrence). A Gaussian distribution was approximated by summing six uniformly distributed samples and properly scaling the sum, resulting in a zero mean and unit variance gaussian random variable, as follows:

$$n_i = \sqrt{2} \sum_{i=1}^{6} (u_i - 0.49999995) \qquad (8)$$

where: n_i is random noise normally distributed with zero mean and unit variance;

u_i is a random number with uniform distribution;

Details on how the numerical values of this equation were obtained will be found in Appendix 2.

The variance or the standard deviation of the distribution can be adjusted if desired by simply multiplying the distribution by the square root of the desired variance or by the desired standard deviation respectively.

B. Correlated Noise

The convolution of a spread function (e.g. the scanning of an aperture) across an array of uncorrelated noise created using the method described above will result in low-pass filtering of the uncorrelated noise, thus creating correlated noise. However, instead of convolving a spatial two-dimensional low-pass filter across the array, we operated in the frequency domain for simplicity.

The uncorrelated noise array is transformed in the frequency domain using a two-dimensional Fourier transform. The resulting array is then multiplied point by point by a two-dimensional gaus function to reduce the high-frequency

noise.

The gaus function is radially symmetrical and of the form:

$$Q(i,j) = gaus\left(\frac{i}{128}\right) gaus\left(\frac{j}{128}\right) \qquad (9)$$

where i and j are pixel coordinates. The scaling factors were set to 128 to ensure a significant amount of correlation.

The computer program in Appendix 3 was written to create the appropriate arrays of uncorrelated and correlated noise, and add them to the original images. The program prompts the user to specify the desired standard deviation of the noise distribution. For the purpose of this study, standard deviations of 15 and 30 (in gray level digital values) were used to evaluate the impact of two different noise levels.

3. CORRUPTION OF IMAGES

In this study, both the uncorrelated and correlated noise cases were considered, and two levels of combined additive and multiplicative noise were used to represent low and high levels of corruption by using different standard deviations in the noise distributions. Figure 5 illustrates the different

treatments used for the initial corruption of images for this study. Noise levels that were added were significant when compared to the noise inherent to the initial image, therefore it may be assumed that the initial images were free of noise to start with.

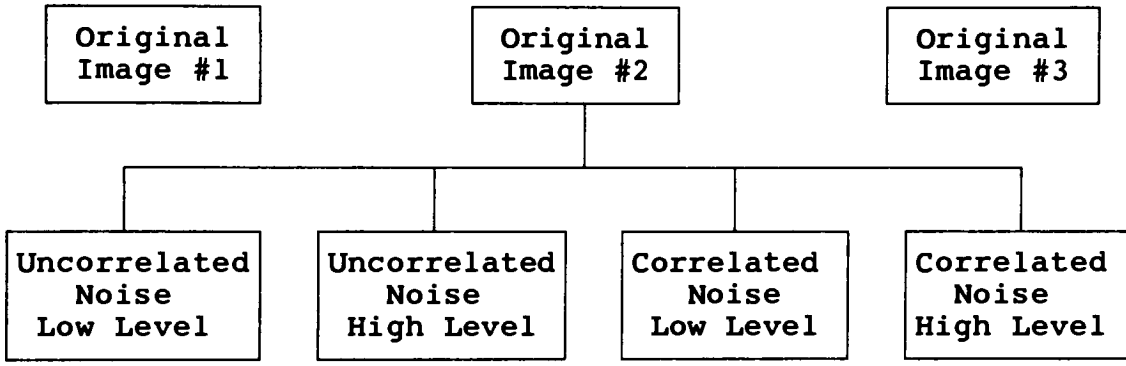


Figure 5. : Noise corruption: Initial treatments

Each of these three images was corrupted according to the above treatments, resulting in twelve initial noisy images.

Representation of multiplicative noise may be achieved by scaling random gaussian noise as a function of the input signal, and then add the result to the signal itself.[29] This can be easily achieved through the use of a look-up table (different look-up tables may be used based on specific situations), or with a signal dependant function as was the case for this study. A linear approximation was used in this

study, in keeping with experimental results obtained with granularity studies, as discussed in the introduction section. Thus,

$$z_{ij} = x_{ij} + k(x_{ij})\ w_{ij} \qquad (10)$$

where z_{ij} is the gray level of the corrupted image at pixel location i,j

x_{ij} is the gray level in the original image

w_{ij} is the noise value from the corresponding pixel in the noise array.

$k(x_{ij})$ is the scaling factor, function of x_{ij}

The scaling factor $k(x_{ij})$ varied linearly between the values of 0.85 to 1.15 as the input signal varied from dark (gray level 0) to bright (gray level 255) to represent this condition.

Random noise arrays with standard deviations of 15 and 30 were used in the study to represent the two different levels of signal corruption.

The above was accomplished with the use of the computer program in Appendix 3, and a sample of the corrupted images is shown in Appendix 5.

4. CREATION OF NOISE FILTERING ALGORITHMS

The computer program in Appendix 6 was written to implement the required filtering algorithms. This program was written in FORTRAN language.

Each noisy image was then filtered with the following algorithms:

a. original Tomita/Tsuji as described earlier;

b. original Nagao/Matsuyama as described earlier;

c. modified Tomita/Tsuji, where the R/d_2 statistic is used in lieu of the variance in order to determine the most homogeneous region of the neighborhood.

d. modified Nagao/Matsuyama, with the same substitution as in the modified Tomita/Tsuji algorithm.

The images to be filtered were read from disk memory (in binary format) and stored in core memory as INTEGER*2 type data for filtering purposes. The images were scanned in rows and the neighborhoods surrounding each pixel of interest were defined according to the appropriate filtering method as described in the introduction section.

The mean and variance of gray levels in all neighborhoods around the pixel of interest were computed and a successive comparison of variances determined which neighborhood had the

lowest variance. The pixel of interest was then assigned the corresponding mean. A new image was thus created and used as output.

In the modified algorithms, the range for each neighborhood was computed using a successive comparison of gray levels in a given neighborhood, and then successive comparison between neighborhoods determined which one had the lowest range. The mean of the corresponding neighborhood was computed and assigned to the pixel of interest.

In all the algorithms considered, a 5 by 5 pixel mask is used around the pixel of interest. If the pixel of interest is located in the two rows or two columns at the edges of the image, all calculations required for filtering were performed over the resulting truncated neighborhoods, provided that at least half the pixels of the full neighborhoods are present.

The Nagao and Matsuyama filter is iterative and in theory would be applied until no more changes occurred in the image. However the results published by these authors[17] show that several iterations of the filter did yield acceptable results. Therefore, five iterations were used in this study in order to keep computing time to a reasonable level.

The twelve images initially corrupted by noise were filtered with the four algorithms discussed above, resulting in a total of 48 final images. The output images were stored in disk memory on the VAX computer. Figure 6 depicts the treatments used in the filtering process.

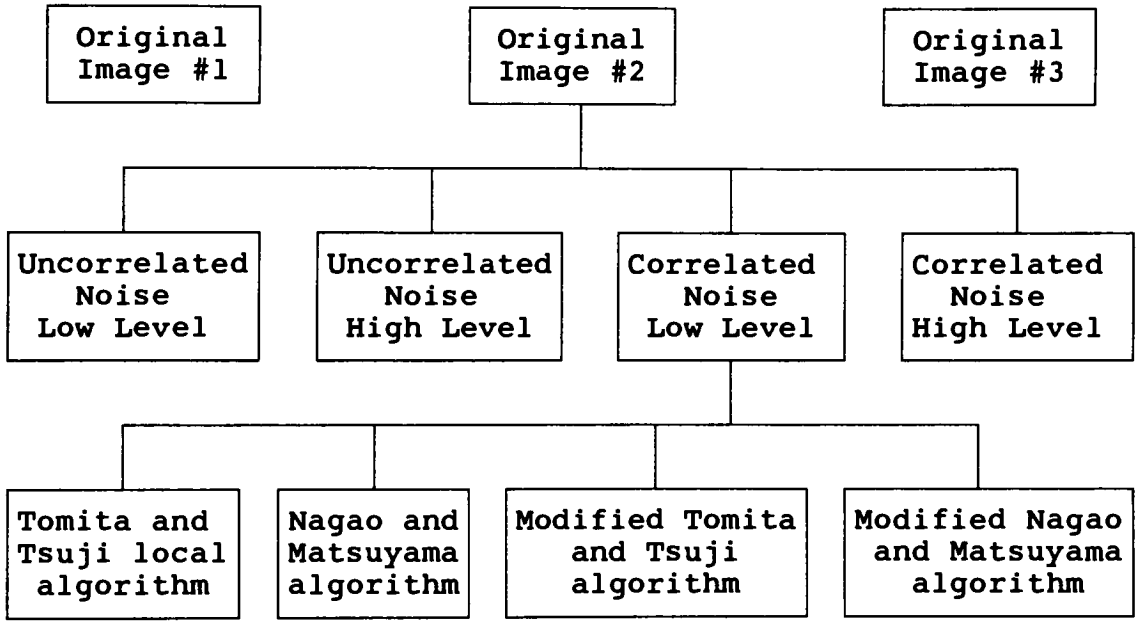


Figure 6. Noise Filtering Treatments

A sample of the resulting filtered images is shown in Appendix 7.

5. OBTAIN HARD COPIES OF IMAGES

After filtering, the digital images were displayed on the Gould DeAnza IP6400 Image Processor located in RIT's Center for Imaging Science. They were transferred from RIT's VAX disk memory to the DeAnza Image Processor via magnetic computer tape and floppy disks.

Hard copies of all images were made with Plus-X pan film using a Dunn CRT-camera system connected to the DeAnza system, and were printed on Polycontrast Rapid II RC paper, five inch square in size.

Some errors were introduced during the photographic reproduction process that may affect the subjective evaluation described later. These include film and paper photographic noise, contrast differences between the image viewed on a monitor and its hard copy, non-linear tone reproduction, some loss of sharpness or fine details due to the optical transfer function of the lenses used, and distortion due to film stretching, to name only a few. It is assumed for the purpose of this study that the errors listed above are either small when compared to the effects to be evaluated and are assumed to be negligible, or are constant for all hard copies generated and will not adversely affect the results. For example, photographic noise levels of the film and paper are

small when compared to the high levels of noise artificially generated for this study, and are considered negligible. The tone reproduction process is constant for all hard copies created, and will be assumed to be linear. Since 512 pixels are imaged over at least 20 mm on the film, the pixel frequency on the film is 25 pixels/mm or less; every pixel will be imaged, thus preserving small details. It is also assumed that no additional artifacts are introduced in the process. Distortion of the images due to the lenses used or to the stretching of the film is assumed constant for all hard copies and will not adversely affect the evaluation results.

6. OBJECTIVE ANALYSIS

A. Noise Reduction

An objective measurement of the effectiveness of each algorithm at noise smoothing was carried out by comparing the variance of gray levels in uniform areas of the gray scale after filtering with the four algorithms.

B. Time Savings

Computational efficiency was determined by comparing the computer time required to filter the images using the

different algorithms.

7. SUBJECTIVE ANALYSIS BY PAIRED COMPARISON

A subjective evaluation of the performance of the noise filtering algorithms was carried out over twelve series of images. Each series comprised one of the twelve noisy images produced earlier and its four associated filtered images.

The five images in each series were compared in pairs as follows:

Image	A	vs.	Image B
	A	vs.	C
	A	vs.	D
	A	vs.	E
	B	vs.	C
	B	vs.	D
	B	vs.	E
	C	vs.	D
	C	vs.	E
	D	vs.	E

Figure 7. Paired Comparison Process

Observers selected the image they perceived to be the best in each pair based on four evaluation criteria, and rated the relative difference between the two images on a scale from zero (no perceived difference) to ten (extreme difference).

The four criteria used in this paired comparison were:

a. effectiveness at noise smoothing - the reduction in noise variance in a flat region of the image. Twelve sets of ratings (one per image series) were produced for this criterion;

b. preservation of subtle details and linear features - ability to retain highly distinguishable subtle details and line features (piers, roads, etc.). Only eight sets of ratings were produced for this criterion, as there were no subtle details or line features to evaluate in the four gray scale images;

c. immunity from shape distortion - the algorithms may create significant distortions due to the directional subregion averaging, as well as artifacts (presence of a structure in the enhanced image with no ground truth or basis for existence other than being artificially induced by a computer algorithmic process).[25] Twelve sets of ratings were produced for this criterion; and

d. retention of step edges and sharpening of ramp edges - highly desirable in both image smoothing and segmentation. Twelve sets of ratings were also produced for this criterion.

The paired comparison took place in a standard light booth with daylight illumination (CIE illuminant D_{65}). The

images were set on a table top at a viewing distance of approximately 18 inches, and a total of 15 observers participated in the evaluation.

8. DATA REDUCTION AND ANALYSIS

Forty-four sets of ten ratings were collected from each observer during the paired comparison. The first phase of data reduction consisted in placing the five images of each set on a linear scale to visualize their relative order and their dispersion after filtering had taken place. Matrix algebra was used in this process as shown in the Results chapter.

The second phase involved a normalization of this data using a method suggested by Granger,[32] as conclusive inferences cannot be drawn from this early data. The fifteen observers involved in this study each assigned ratings on how different the two images were in each pair based on their unique personal values and judgement. A rating of 9 given by one observer may correspond to a rating of 6 to another observer or 5 to yet another one. In addition, one observer may have assigned ratings from 2 to 7 throughout the whole paired comparison process, whereas another one used the full range from 0 to 10. In order to achieve better correlation between the data sets and to draw more precise inferences, a normali-

zation was performed by first establishing a common scale for all observers. A simple linear regression technique was used for this purpose.

The normalized data sets were next used to compute average statistical distances for each image on the above linear scale, in each of the forty-four data sets. These distances were in turn used to verify the initial hypothesis expressed in the introduction section. If two different algorithms are to produce images of equivalent quality when applied to the same noisy image, then it should not be possible to differentiate between their average statistical distances. Hypothesis testing concerning two means was therefore carried out using the average statistical distances and their associated standard deviation.

Detailed data reduction techniques are explained in the following chapter.

III. RESULTS

1. OBJECTIVE ANALYSIS

A. Noise Reduction

An analysis of two steps of the gray scale image (namely the steps with digital gray levels of 85 and 170) was carried out and the average gray level and standard deviation were computed to objectively assess the effectiveness of the local filters at reducing noise. Results are presented in Table 2.

B. Computer Time Savings

In order to evaluate computer time savings, each noise filtering algorithm was run separately over a noisy image to measure individual running times. The CPU run time was used for this analysis, and was recorded from the logging off message at the end of each run. The CPU time spent in overhead tasks (eg. logging on, accessing disks and files) common to all filtering runs was 0.186 CPU minutes and was subtracted from the total run times. This allowed the measuring of the net run time of the filter. The net results are presented in Table 3.

TABLE 2

OBJECTIVE EVALUATION OF NOISE REDUCTION THROUGH MEASUREMENTS IN TWO STEPS OF GRAY SCALE IMAGE

GRAY SCALE	STEP 6			STEP 11		
	MEAN LEVEL	STD. DEV'N	DELTA %	MEAN LEVEL	STD. DEV'N	DELTA %
Clean Gray Scale	85	0.0	N/A	170	0.0	N/A
Noisy with Low Level Uncorrelated Noise	85	12.93	0.0%	169	12.91	0.0%
Tomita/Tsuji filter	85	4.59	-64.5%	170	4.42	-65.8%
Modified Tomita/Tsuji	84	4.85	-62.5%	169	4.74	-63.3%
Nagao/Matsuyama filter	83	2.66	-79.4%	168	2.45	-81.0%
Modified Nagao/Matsuyama	85	4.11	-68.2%	169	4.04	-68.7%
Noisy with High Level Uncorrelated Noise	85	25.84	0.0%	169	25.77	0.0%
Tomita/Tsuji filter	86	9.17	-64.5%	170	8.87	-65.6%
Modified Tomita/Tsuji	84	9.66	-62.6%	168	9.52	-63.1%
Nagao/Matsuyama filter	84	5.41	-79.1%	168	5.01	-80.6%
Modified Nagao/Matsuyama	85	8.23	-68.2%	169	8.00	-69.0%
Noisy with Low Level Correlated Noise	85	13.00	0.0%	169	12.82	0.0%
Tomita/Tsuji filter	85	8.46	-34.9%	169	8.15	-36.4%
Modified Tomita/Tsuji	85	9.10	-30.0%	168	8.86	-30.9%
Nagao/Matsuyama filter	84	5.53	-57.5%	168	5.09	-60.3%
Modified Nagao/Matsuyama	86	8.02	-38.3%	169	7.27	-43.3%
Noisy with High Level Correlated Noise	85	25.99	0.0%	169	25.60	0.0%
Tomita/Tsuji filter	86	16.95	-34.8%	169	16.28	-36.4%
Modified Tomita/Tsuji	85	18.25	-29.8%	168	17.66	-31.0%
Nagao/Matsuyama filter	86	11.34	-56.4%	168	10.26	-59.9%
Modified Nagao/Matsuyama	87	16.04	-38.3%	169	14.54	-43.2%

TABLE 3 - COMPUTER TIME SAVINGS

Computer time required to run algorithms:	Net CPU Times	Time Savings
Tomita/Tsuji	1.537 min	N/A
Modified Tomita/Tsuji	0.965 min	37.2 %
Nagao/Matsuyama	12.276 min	N/A
Modified Nagao/Matsuyama	8.927 min	27.3 %

2. SUBJECTIVE ANALYSIS

A. Preliminary Results

Every series of images used in the paired comparison process consisted of one noisy image and four filtered images (as depicted in figure 6). All observers selected the best image from each pair (as listed in figure 7) and assigned a rating to indicate how different they felt the two images were. A sample of this data is shown in Appendix 8 in the form of an evaluation sheet completed by one of the observers.

B. Raw Statistical Distances

From the above ten ratings, it is now possible to determine the relative ranking of the images on a linear scale ranging from minus ten to plus ten as shown in figure 8.

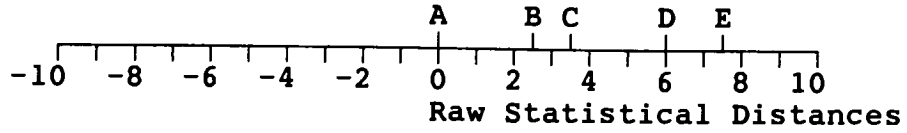


Figure 8. Linear Scale for Relative Ranking of Images in a Series.

In the above figure, A represents one of the five images and is arbitrarily set to zero for reference purposes. When located on the positive axis on this scale, the other images (labelled B, C, D and E) are perceived as being better than image A for the criterion under evaluation. Conversely, negative values indicate that an image is perceived as being worse than image A. Figure 8 depicts what one would expect when rating the algorithms' noise filtering ability, with image A as the noisy image and algorithm E as being the best.

No system of units is associated with this linear scale. Its values are derived from the same scale of zero to ten used during the paired comparison, and represent the relative perceived differences in image quality between images of a given series (hence the relative efficiency of each filter).

The values for A to E on this linear scale will be referred to as raw statistical distances. They are obtained by using image A as a reference point (i.e. arbitrarily set to zero) and by solving the following matrix system:

$$
\begin{vmatrix} A-B \\ A-C \\ A-D \\ A-E \\ B-C \\ B-D \\ B-E \\ C-D \\ C-E \\ D-E \end{vmatrix}
=
\begin{vmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \\ 1 & -1 & 0 & 0 \\ 1 & 0 & -1 & 0 \\ 1 & 0 & 0 & -1 \\ 0 & 1 & -1 & 0 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & -1 \end{vmatrix}
\bullet
\begin{vmatrix} \\ B \\ \\ C \\ \\ D \\ \\ E \\ \\ \end{vmatrix}
$$

J M S

Figure 9. Matrix system used to compute raw statistical distances (where A=0)

Matrix J contains one observer's ratings for the ten pairs of images in a given series. Let us say that an observer has chosen image B as being the best in the first pair (between images A and B), and has given it a rating of 4. The expression "A minus B" in the matrix is then assigned the value minus 4. In keeping with the sign convention defined above, a negative sign is thus assigned to all ratings where the second image in the pair was chosen as the best. Matrix M contains fixed coefficients applying to this particular case, and matrix S contains the raw statistical distances for images B, C, D and E. Image A is arbitrarily set to zero, and by solving for matrix S, one obtains raw relative distances with respect to image A for one observer. The same calculations are then performed for the 15 observers for a given evaluation criterion in a given series of images.

From figure 9, we have:

$$| M | \bullet | S | = | J | \qquad (11)$$

One solves for S as follows:

$$|M^T M| \bullet |S| = |M^T| \bullet |J| \qquad (12)$$

$$|M^T M|^{-1} \bullet |M^T M| \bullet |S| = |M^T M|^{-1} \bullet |M^T| \bullet |J| \qquad (13)$$

$$| S | = | M^T M |^{-1} \bullet | M^T | \bullet | J | \qquad (14)$$

Using matrix algebra, the solution for this particular situation is as follows:

$$\text{Let } w = AB + BC + BD + BE \qquad (15)$$

$$x = AC - BC + CD + CE \qquad (16)$$

$$y = AD - BD - CD + DE \qquad (17)$$

$$z = AE - BE - CE - DE \qquad (18)$$

where AB, AC, etc. represent the observers ratings; then:

$$A = 0 \quad \text{(arbitrarily set)} \qquad (19)$$

$$B = 0.4\,w + 0.2\,x + 0.2\,y + 0.2\,z \qquad (20)$$

$$C = 0.2\,w + 0.4\,x + 0.2\,y + 0.2\,z \qquad (21)$$

$$D = 0.2\,w + 0.2\,x + 0.4\,y + 0.2\,z \qquad (22)$$

$$E = 0.2\,w + 0.2\,x + 0.2\,y + 0.4\,z \qquad (23)$$

The raw distances were calculated with a programmable hand-held calculator, and the values were written to an ASCII

computer file for further data reduction.

The computer program in Appendix 9 was written to perform all remaining data reduction, including the hypothesis testing. A sample output from that program can be found in Appendix 10, with raw statistical distances shown under that heading.

C. Adjusted Statistical Distances

In order to test the hypothesis that the modified algorithms produce images of equivalent quality to those produced by the original algorithms, one must compute the average statistical distances from all observers as well as their standard deviation, as explained in the next section.

In an ideal case, all observers would assign ratings based on the same set of personal values and biases. By plotting one observer's raw statistical distances against those of another, one would expect to see a directly proportional relationship with a high degree of correlation as shown in figure 10. If this were the case, one could average the statistical distances among all observers and expect a relatively narrow standard distribution, and proceed to hypothesis testing.

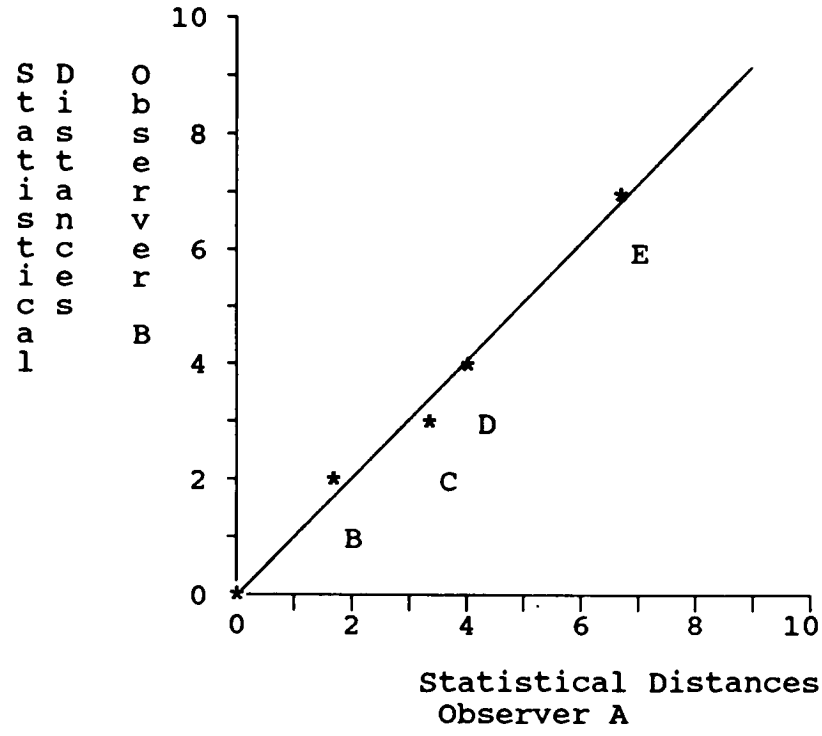


Figure 10. Plot of Raw Statistical Distances Between Two Ideal Observers

In the paired evaluation such as the one performed in this study, however, each observer made a personal judgement of a quantitative nature based on a different set of personal values and biases. As mentioned earlier, some observers used higher or lower ratings than others to describe the difference between the same two images, or used a narrower range of ratings throughout the whole process (eg. from 2 to 6 for one observer, and from 0 to 10 for another).

In this study, plots of one observer's data to that of another were not expected to show the relationship of figure

10. A typical example of what was expected is shown in figure 11 where the slope of the linear relationship is different than one and where the y-intercept is non-zero. It was assumed that the relationship would still remain linear.

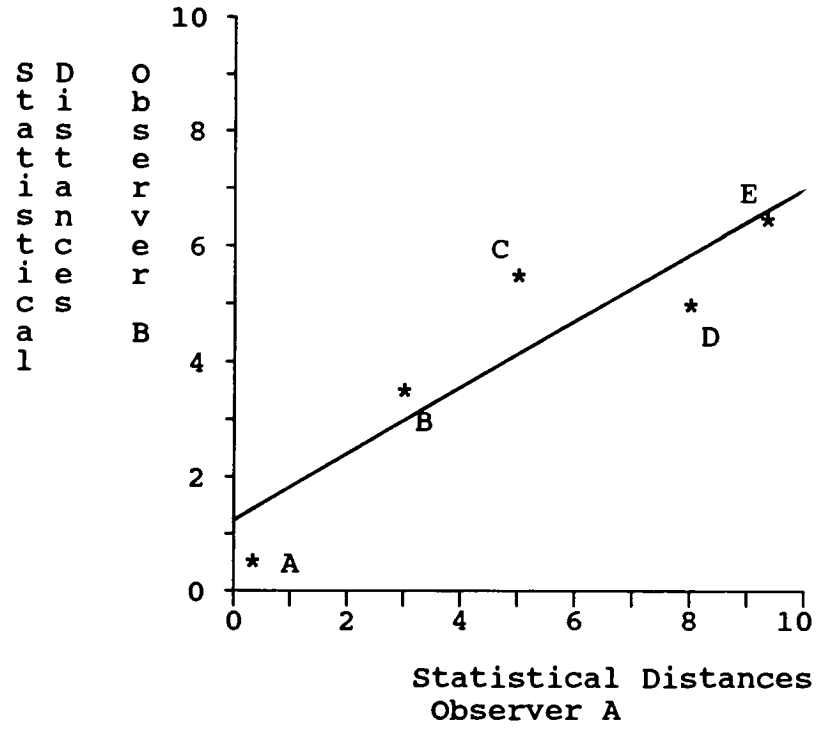


Figure 11. Typical Plot of Raw Statistical Distances Between Two Actual Observers

If averaged at this stage, the raw statistical distances would have a wider standard deviation than with ideal observers and hypothesis testing would lead to less precise conclusions. In order to derive more accurate inferences, the raw statistical distances of all observers were normalized to the distances of a reference observer. The observer whose

data showed the most correlation with other observers' data was selected as the reference observer. The correlation statistics used in this selection process are shown in Appendix 10. All observers' raw statistical distances were adjusted with a simple linear regression calculation to obtain a directly proportional relationship with the reference observer's data (i.e. slope of 1 and zero intercept). Appendix 10 shows the results of this process under the heading "Adjusted Distances".

After this normalization, the average value of the 15 adjusted distances was computed, as well as their standard deviation. They are shown under the heading "Mean Adjusted Distances" in Appendix 10. A normal distribution was assumed for the adjusted distances. If one of the 15 values was found to be outside the limits of three standard deviations on either side of the mean (i.e. where 99.9% of a normally distributed population should be) for any of the adjusted distances, then all data from that observer were rejected and new averages and standard deviations were computed with the remaining data points. The three standard deviation limits are listed with the mean adjusted distances in Appendix 10. This was repeated until all data points were within three standard deviations of the mean. From a total of 44 series of mean adjusted distances, one data set was rejected from 12 series, two data sets were rejected from 3 series, three data

sets were rejected from 1 series and four data sets were rejected from 2 series. Some observers were repeat offenders in these rejections; it was therefore assumed that the instructions for the evaluation were not totally understood by these observers.

Some data sets were negatively correlated when compared to the reference data set. In some of these cases, however, the magnitude of the correlation factor was as high as in cases with a strong positive correlation, suggesting that the observer simply selected the opposite images by mistake during the paired comparison. Some observers later confirmed this assumption. Data sets with correlation factors ranging between -0.5 and -1.0 were therefore assumed to be valid and were retained. Data with lower negative correlation factors were eliminated. It was found that almost all the data sets that were eliminated in this way would also have been eliminated as being outside the three standard deviation limits.

The labels A to E were assigned randomly to the five images in each series to maintain the randomness of the process (i.e. to avoid the development of a pattern or of expectations during the paired comparison). The final step of the normalization process was then to shift the mean adjusted distance values to set the noisy image to zero for standardi-

zation between series.

The results of the above data reduction are shown in Tables 4 to 7, where each table lists the shifted mean adjusted distances. A different table covers each of the four evaluation criteria.

The subjective data of Table 4 (in image 1) generally points to the same trends as the objective data of Table 2. Higher mean distance values for image 1 in Table 4 (which indicate better noise smoothing ability of the algorithm) correlate with lower standard deviation values in Table 2 (which indicate the noise level present in the filtered image). This correlation provides added confidence in the subjective evaluation process used in this study.

TABLE 4

SUBJECTIVE PAIRED COMPARISON
MEAN ADJUSTED DISTANCES
NOISE SMOOTHING ABILITY

IMAGE EVALUATED	IMAGE 1: LOW INFO CONTENT		IMAGE 2: MEDIUM INFO CONTENT		IMAGE 3: HIGH INFO CONTENT	
	MEAN DIST.	STD. DEV'N	MEAN DIST.	STD. DEV'N	MEAN DIST.	STD. DEV'N
Noisy with Low Level Uncorrelated Noise	0.00	0.31	0.00	0.54	0.00	0.72
Tomita/Tsuji filter	3.27	0.98	1.79	0.30	2.80	0.61
Modified Tomita/Tsuji	3.97	0.40	1.48	0.29	2.96	0.38
Nagao/Matsuyama filter	5.53	0.70	3.28	0.19	3.99	0.68
Modified Nagao/Matsuyama	4.12	1.21	1.10	0.79	2.61	0.89
Noisy with High Level Uncorrelated Noise	0.00	0.25	0.00	0.59	0.00	0.21
Tomita/Tsuji filter	3.56	0.65	1.61	0.33	2.96	0.54
Modified Tomita/Tsuji	3.09	0.88	1.53	0.57	2.64	0.62
Nagao/Matsuyama filter	5.47	0.95	3.53	0.25	4.58	0.44
Modified Nagao/Matsuyama	3.95	1.35	1.43	1.23	2.99	0.54
Noisy with Low Level Correlated Noise	0.00	0.61	0.00	0.27	0.00	0.53
Tomita/Tsuji filter	3.86	0.91	2.39	0.64	5.35	1.14
Modified Tomita/Tsuji	3.56	0.78	2.21	0.39	5.01	0.89
Nagao/Matsuyama filter	6.87	0.66	4.68	0.34	8.88	1.08
Modified Nagao/Matsuyama	3.76	1.69	2.47	0.66	5.38	1.59
Noisy with High Level Correlated Noise	0.00	0.62	0.00	0.62	0.00	0.16
Tomita/Tsuji filter	4.78	1.20	-2.34	0.26	1.64	0.27
Modified Tomita/Tsuji	4.36	1.16	-0.10	0.72	1.47	0.30
Nagao/Matsuyama filter	8.28	0.42	2.16	0.23	3.01	0.19
Modified Nagao/Matsuyama	4.62	1.87	-0.29	1.18	1.96	0.52

TABLE 5

SUBJECTIVE PAIRED COMPARISON
MEAN ADJUSTED DISTANCES
PRESERVATION OF SUBTLE DETAILS

IMAGE EVALUATED	IMAGE 1: LOW INFO CONTENT		IMAGE 2: MEDIUM INFO CONTENT		IMAGE 3: HIGH INFO CONTENT	
	MEAN DIST.	STD. DEV'N	MEAN DIST.	STD. DEV'N	MEAN DIST.	STD. DEV'N
Noisy with Low Level Uncorrelated Noise	N/A	N/A	0.00	0.29	0.00	0.42
Tomita/Tsuji filter	N/A	N/A	-0.65	0.37	-3.12	0.43
Modified Tomita/Tsuji			-1.31	0.35	-3.86	0.64
Nagao/Matsuyama filter			-3.67	0.27	-7.03	0.63
Modified Nagao/Matsuyama			-2.54	0.42	-5.27	0.55
Noisy with High Level Uncorrelated Noise	N/A	N/A	0.00	0.68	0.00	0.52
Tomita/Tsuji filter	N/A	N/A	-3.24	0.78	-3.25	0.75
Modified Tomita/Tsuji			-3.41	0.73	-3.96	0.79
Nagao/Matsuyama filter			-8.06	0.53	-6.50	0.79
Modified Nagao/Matsuyama			-6.24	0.69	-5.39	0.55
Noisy with Low Level Correlated Noise	N/A	N/A	0.00	0.65	0.00	0.67
Tomita/Tsuji filter	N/A	N/A	-1.65	1.08	-3.38	0.63
Modified Tomita/Tsuji			-2.91	0.60	-3.52	0.81
Nagao/Matsuyama filter			-7.91	0.37	-7.86	0.70
Modified Nagao/Matsuyama			-4.95	0.70	-5.92	0.53
Noisy with High Level Correlated Noise	N/A	N/A	0.00	0.89	0.00	1.93
Tomita/Tsuji filter	N/A	N/A	1.69	0.69	-2.38	1.47
Modified Tomita/Tsuji			-0.45	1.27	-3.40	1.55
Nagao/Matsuyama filter			-5.18	0.73	-8.97	1.42
Modified Nagao/Matsuyama			-3.25	0.83	-6.53	1.19

TABLE 6

SUBJECTIVE PAIRED COMPARISON
MEAN ADJUSTED DISTANCES
IMMUNITY TO SHAPE DISTORTION

IMAGE EVALUATED	IMAGE 1: LOW INFO CONTENT		IMAGE 2: MEDIUM INFO CONTENT		IMAGE 3: HIGH INFO CONTENT	
	MEAN DIST.	STD. DEV'N	MEAN DIST.	STD. DEV'N	MEAN DIST.	STD. DEV'N
Noisy with Low Level Uncorrelated Noise	0.00	1.00	0.00	0.88	0.00	1.49
Tomita/Tsuji filter	-4.67	2.07	-2.58	1.13	-2.96	0.87
Modified Tomita/Tsuji	-3.56	1.31	-3.72	1.66	-4.46	1.69
Nagao/Matsuyama filter	-8.87	1.14	-9.35	1.39	-10.37	1.23
Modified Nagao/Matsuyama	-6.62	1.95	-8.53	1.37	-8.48	1.54
Noisy with High Level Uncorrelated Noise	0.00	0.96	0.00	0.69	0.00	0.69
Tomita/Tsuji filter	-2.00	0.93	-3.32	1.02	-0.86	0.58
Modified Tomita/Tsuji	-1.17	0.94	-3.75	0.52	-1.99	0.52
Nagao/Matsuyama filter	-4.43	0.63	-6.91	1.10	-5.59	0.70
Modified Nagao/Matsuyama	-3.48	0.93	-6.58	0.80	-3.67	0.76
Noisy with Low Level Correlated Noise	0.00	0.32	0.00	2.21	0.00	0.96
Tomita/Tsuji filter	-1.74	0.55	-3.10	3.04	-1.24	0.76
Modified Tomita/Tsuji	-1.04	0.50	-4.30	1.27	-0.91	0.90
Nagao/Matsuyama filter	-3.99	0.48	-10.17	2.00	-4.98	0.48
Modified Nagao/Matsuyama	-2.94	0.88	-8.56	2.31	-3.17	0.81
Noisy with High Level Correlated Noise	0.00	1.37	0.00	0.35	0.00	0.69
Tomita/Tsuji filter	-2.11	1.48	0.93	0.53	-0.26	0.46
Modified Tomita/Tsuji	-1.44	1.63	-0.30	0.67	-0.50	0.38
Nagao/Matsuyama filter	-5.59	1.65	-1.62	0.63	-2.71	0.30
Modified Nagao/Matsuyama	-7.09	1.34	-1.48	0.47	-1.70	0.36

TABLE 7

SUBJECTIVE PAIRED COMPARISON
MEAN ADJUSTED DISTANCES
PRESERVATION OF SHARP EDGES

IMAGE EVALUATED	IMAGE 1: LOW INFO CONTENT		IMAGE 2: MEDIUM INFO CONTENT		IMAGE 3: HIGH INFO CONTENT	
	MEAN DIST.	STD. DEV'N	MEAN DIST.	STD. DEV'N	MEAN DIST.	STD. DEV'N
Noisy with Low Level Uncorrelated Noise	0.00	0.69	0.00	0.43	0.00	1.28
Tomita/Tsuji filter	-2.20	1.63	1.59	0.88	1.13	1.56
Modified Tomita/Tsuji	-1.01	1.09	1.87	0.74	2.84	0.44
Nagao/Matsuyama filter	-3.35	0.84	2.16	1.28	-0.46	1.08
Modified Nagao/Matsuyama	-2.00	0.75	2.32	0.44	4.38	1.16
Noisy with High Level Uncorrelated Noise	0.00	0.60	0.00	0.67	0.00	2.77
Tomita/Tsuji filter	-1.26	0.78	2.14	1.68	3.00	2.65
Modified Tomita/Tsuji	-0.52	1.07	2.45	1.17	5.19	1.17
Nagao/Matsuyama filter	-2.82	0.51	3.86	1.75	0.47	2.11
Modified Nagao/Matsuyama	-1.25	0.86	3.80	1.49	5.31	2.30
Noisy with Low Level Correlated Noise	0.00	0.37	0.00	1.00	0.00	1.78
Tomita/Tsuji filter	-3.78	1.89	2.92	1.62	0.65	1.05
Modified Tomita/Tsuji	-1.33	1.06	2.16	1.35	1.99	1.17
Nagao/Matsuyama filter	-6.24	0.66	2.64	2.57	-0.38	1.82
Modified Nagao/Matsuyama	-3.64	0.56	3.75	1.93	2.51	1.40
Noisy with High Level Correlated Noise	0.00	1.35	0.00	0.78	0.00	1.90
Tomita/Tsuji filter	-2.78	1.81	-1.97	0.21	0.43	1.01
Modified Tomita/Tsuji	-2.09	1.73	0.44	1.17	1.32	0.41
Nagao/Matsuyama filter	-6.79	1.59	1.28	1.89	-2.26	1.16
Modified Nagao/Matsuyama	-6.06	3.11	1.09	1.41	1.26	1.32

D. Hypothesis Testing

The underlying hypothesis of this study was that the substitution of the range for the variance as a decision criterion in the modified algorithms would yield similar image quality when the final images are compared. To verify this, hypothesis testing concerning the difference between two means was carried out to determine whether any difference between the mean adjusted distances from two algorithms (our measure of image quality) is significant or whether it may be attributed to chance.

Using samples of sizes n_1 and n_2 from two normal distributions with means μ_1 and μ_2 and variances σ_1^2 and σ_2^2, we can test the null hypothesis H_0: $\mu_1 - \mu_2 = \delta$, where δ is a specified constant, against the alternate H_1: $\mu_1 - \mu_2 \neq \delta$. In this study, we want to test that the means are equal (corresponding to equal image quality), where δ equals zero. We then have:

$$\text{Null Hypothesis } H_0: \quad \mu_1 = \mu_2 \tag{24}$$

$$\text{Alternate } H_1: \quad \mu_1 \neq \mu_2 \tag{25}$$

The test depends on the difference between the sample means $\bar{x}_1$ and $\bar{x}_2$, and if both samples come from normal populations with known variances, it can be based on the

statistic

$$z = \frac{(\overline{x_1} - \overline{x_2}) - \delta}{\sqrt{\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}}} \qquad (26)$$

The population means need not be known to use (26), but since the population variances are also unknown, the above statistic could not be used. The population variances could have been approximated with the variance of the respective samples had the sample sizes been large enough; however the sample sizes in this study were considered too small to satisfy that condition. A suitable t-statistic was therefore used. It was assumed that both populations are normal and that $\sigma_1^2 = \sigma_2^2$. Based on these assumptions, it can be shown, when $n_1 = n_2 = n$, that the test can be based on the statistic

$$t = \sqrt{n}\,\frac{(\overline{x_1} - \overline{x_2})}{\sqrt{s_1^2 + s_2^2}} \qquad (27)$$

The level of significance for the test was $\alpha = 0.05$, where α corresponds to the risk of rejecting a hypothesis even though it is true. The critical value $t_{\alpha/2}$ for this two-tail test, $t_{0.025}$ for $(n_1 + n_2 - 2)$ degrees of freedom, was obtained from statistical tables, and the null hypothesis was rejected

when $|t| > t_{0.025}$. Note that $t_{\alpha/2}$ is dependent upon the sample size n, and its value changed when data sets were rejected as explained earlier.

Appendix 10 shows a sample of the test results under the heading "Hypothesis Testing". The value of $t_{0.025}$ and the corresponding number of observers n are indicated first, followed by the value of the t statistic (as per (27) above) for several pairs of images.

The above two-tail t-test was repeated for the 44 sets of ratings. The results are compiled in Table 8, in which the numerical values correspond to the t-statistics computed above, providing an indication of the differences between the means. A minus sign indicates that the first image of the pair (as listed in the legend block) was perceived to be the best image in the paired comparison; a plus sign indicates that the second image was preferred. Finally, an equal sign appears in cases where the Null hypothesis could not be rejected, and N/A indicates a criterion that was not evaluated.

TABLE 8

HYPOTHESIS TESTING RESULTS

PAIR	IMAGE 1				IMAGE 2				IMAGE 3			
	LoLevel Uncor'd Noise	HiLevel Uncor'd Noise	LoLevel Corrl'd Noise	HiLevel Corrl'd Noise	LoLevel Uncor'd Noise	HiLevel Uncor'd Noise	LoLevel Corrl'd Noise	HiLevel Corrl'd Noise	LoLevel Uncor'd Noise	HiLevel Uncor'd Noise	LoLevel Corrl'd Noise	HiLevel Corrl'd Noise
	EVALUATION CRITERION: NOISE SMOOTHING											
A	+ 14.0	+ 19.9	+ 13.7	+ 13.7	+ 11.3	+ 9.2	+ 13.3	13.5	+ 10.7	+ 19.2	+ 15.9	+ 19.4
B	+ 30.3	+ 13.1	+ 13.9	+ 12.8	+ 9.4	+ 7.2	+ 18.1	=	+ 13.2	+ 15.2	+ 18.1	+ 16.3
C	+ 27.9	+ 21.6	+ 29.7	+ 43.1	+ 22.3	+ 21.2	+ 41.7	+ 12.7	+ 14.6	+ 35.3	+ 27.7	+ 45.1
D	+ 12.8	+ 11.1	+ 8.1	+ 9.1	+ 4.5	+ 4.1	+ 13.4	=	+ 8.2	+ 19.5	+ 12.0	+ 13.4
E	+ 5.8	+ 6.4	+ 10.4	+ 10.7	+ 16.4	+ 17.9	+ 12.2	+ 49.7	+ 4.7	+ 8.8	+ 8.4	+ 15.3
F	=	=	=	=	2.9	=	=	+ 11.3	=	=	=	=
G	3.9	3.6	6.6	- 7.4	10.4	6.5	11.5	7.9	4.4	8.6	6.8	7.0
H	=	+ 2.1	=	=	=	=	=	=	=	=	=	+ 3.0
	EVALUATION CRITERION: PRESERVATION OF SUBTLE DETAILS											
A	N/A	N/A	N/A	N/A	5.1	12.1	5.1	+ 5.8	20.3	13.9	14.2	3.8
B	N/A	N/A	N/A	N/A	10.7	13.3	12.7	=	19.7	16.3	12.9	5.3
C	N/A	N/A	N/A	N/A	34.6	36.2	41.8	17.5	36.1	26.6	31.5	14.5
D	N/A	N/A	N/A	N/A	18.7	25.0	20.1	10.3	29.7	27.6	26.8	11.1
E	N/A	N/A	N/A	N/A	24.5	19.7	21.3	26.5	19.4	11.6	18.5	12.5
F	N/A	N/A	N/A	N/A	- 4.8	=	- 3.4	5.7	3.8	- 2.5	=	=
G	N/A	N/A	N/A	N/A	+ 8.5	+ 8.1	+ 14.5	+ 6.7	+ 8.2	+ 4.5	+ 8.6	+ 5.1
H	N/A	N/A	N/A	N/A	8.5	10.9	8.5	7.2	6.5	5.8	9.6	6.2
	EVALUATION CRITERION: IMMUNITY TO SHAPE DISTORTION											
A	7.9	5.8	10.2	- 4.0	7.0	10.1	3.2	+ 5.7	6.4	3.6	3.9	=
B	8.4	3.4	- 6.5	- 2.6	13.1	16.3	6.6	=	7.4	8.6	- 2.7	2.4
C	22.6	14.9	- 25.9	10.1	- 22.0	19.9	13.3	8.7	25.2	21.3	17.9	13.5
D	11.7	- 10.0	11.8	14.3	20.3	23.3	10.4	9.8	14.8	13.4	9.7	8.2
E	6.9	8.4	11.5	6.1	14.6	9.0	7.6	12.0	25.1	19.6	16.1	16.8
F	=	+ 2.4	+ 3.5	=	3.4	=	=	5.6	3.0	- 5.5	=	=
G	+ 3.9	+ 3.3	+ 4.0	2.7	=	=	+ 2.1	=	+ 8.7	+ 7.0	+ 7.5	+ 8.0
H	5.0	6.7	7.0	10.4	12.2	11.1	6.2	5.6	6.6	6.9	7.2	8.5
	EVALUATION CRITERION: PRESERVATION OF SHARP EDGES											
A	4.8	4.8	7.4	4.5	+ 6.7	+ 4.8	+ 5.1	9.7	=	+ 2.7	+ 3.1	=
B	3.0	=	- 4.4	- 3.4	+ 10.3	+ 7.1	+ 6.2	=	+ 7.0	+ 6.0	=	+ 2.4
C	11.9	13.3	- 30.9	11.7	+ 8.0	+ 9.9	+ 5.9	+ 2.1	=	=	=	3.7
D	7.6	4.5	- 20.3	6.5	+ 14.5	+ 10.6	+ 10.0	+ 3.4	+ 8.4	+ 5.1	+ 3.7	=
E	2.4	6.3	- 4.6	6.0	=	+ 2.3	=	+ 8.9	- 2.8	- 2.6	=	- 6.3
F	- 2.3	+ 2.1	+ 4.2	=		=		+ 8.0	+ 3.5	+ 2.6	+ 2.8	+ 2.9
G	+ 4.6	+ 5.9	+ 11.2	=		=	=	=	+ 10.1	+ 5.4	+ 4.2	+ 7.2
H	2.9	=	7.2	4.0		+ 3.2	+ 3.5	=	+ 4.1	=	=	=

LEGEND - PAIRS COMPARED			
A	Noisy	vs	Tomita/Tsuji
B	Noisy	vs	Modified Tomita/Tsuji
C	Noisy	vs	Nagao/Matsuyama
D	Noisy	vs	Modified Nagao/Matsuyama
E	Tomita/Tsuji	vs	Nagao/Matsuyama
F	Tomita/Tsuji	vs	Modified Tomita/Tsuji
G	Nagao/Matsuyama	vs	Modified Nagao/Matsuyama
H	Modified Tomita/Tsuji	vs	Modified Nagao/Matsuyama

IV. DISCUSSION

1. NOISE SMOOTHING

The results of the subjective paired comparison for noise smoothing ability shown in Table 4 correlate strongly with the objective results shown in Table 2. The correlation factors between the statistical distances of image 1 in Table 4 and the standard deviation values of Table 2 are: -.9144, -.9966, -.9723 and -.9688 respectively for the four noise conditions identified in these tables. This strong correlation between the perceived noise smoothing ability of the algorithms and the noise present in images reinforces the confidence in the value of the subjective process used in this study.

The values in Table 2 do not show a strong dependency between the average gray level and the filter's noise smoothing ability. All the filters exhibit similar performance whether in an area of average gray level of 85 or 170.

The substitution of the range for the variance in the Tomita and Tsuji algorithm produced images of comparable quality in most cases for the conditions of the test. Table 8 reveals that in ten cases out of twelve, the null hypothesis could not be rejected; and in the other two cases, one image was slightly poorer, and one was judged better. The same

substitution in the Nagao and Matsuyama algorithm resulted in a significant decrease in noise smoothing ability in all noise conditions.

2. PRESERVATION OF SUBTLE DETAILS

As anticipated, all four algorithms removed subtle details from all the scenes they were used on. The substitution of the range for the variance in the Tomita and Tsuji algorithm resulted in a significant loss of fine details in five cases out of eight. In the other three cases, the null hypothesis could not be rejected. Unexpectedly, the modified version of the Nagao and Matsuyama algorithm was found to be significantly better at preserving subtle details than the original algorithm, in all noise conditions.

3. IMMUNITY TO SHAPE DISTORTION

All the filters introduced artifacts or distorted some linear features in the images when compared to the original image. This effect is inversely dependant upon the difference in gray levels (or contrast) between the features of the image. The original Tomita and Tsuji algorithm showed somewhat better immunity to shape distortion than its modified

version in images of medium and high information contents (in four cases out of twelve), whereas the modified version of the Nagao and Matsuyama algorithm comparatively displayed stronger immunity than in the original algorithm (in eight cases out of twelve).

4. PRESERVATION OF STEP EDGES

The preservation of step edges and the sharpening of ramp edges in the noisy images was more efficient with the modified versions of both algorithms (in six cases with the modified Tomita and Tsuji algorithm, and in seven cases with the modified Nagao and Matsuyama filter).

5. COMPARISON WITH PREVIOUS STUDIES

It was reported by Nagao and Matsuyama[17] that their algorithm, being more directional in nature, was more adept at retaining edges than its predecessors (such as Tomita and Tsuji). However since the region over which the averaging is performed in the filtering process is smaller (five or seven pixels as opposed to nine), smoothing is done via an iterative process where the procedure is repeated until there are no more significant changes in the gray levels. They had used

the algorithm on images that represented simple shapes and had fairly large differences in pixel gray levels between boundaries. In this study, the images displayed a lower contrast between features and only five iterations of the filter were performed. The results obtained in this study only support the claim for increased noise smoothing ability when compared to the Tomita and Tsuji algorithm. The iterative process, as originally proposed, did not result in the preservation of sharp edges or in immunity to shape distortion as expected. The results would have been even worse had more iterations been performed. The Nagao and Matsuyama algorithm is therefore not recommended for application on continuous tone, low contrast images.

6. EXPERIMENTAL ERROR

The data collected during this study was highly subject to experimental error due to the different cultural background and experiences of each observer. However, the normalization technique used to adjust raw statistical distances was shown to be effective at removing most of this type of error, as supported by the strong correlation between subjective and objective results.

V. CONCLUSION

1. CONCLUSIONS

The hypothesis behind this research was that the substitution of the statistical range for the variance as a decision criterion in local filtering algorithms would yield comparable results, and in less time.

In the course of this project, a subjective paired comparison was performed, and a technique of normalization of subjective data between 15 observers was introduced prior to hypothesis testing. This technique resulted in strong correlation between subjective and objective data, and is considered valid for a subjective paired comparison process.

Given these manipulations and the condition of the test, it is concluded that the substitution of the statistical range for the variance will lead to:

a. somewhat comparable noise smoothing ability for the Tomita and Tsuji algorithm, but adverse effects with the Nagao and Matsuyama algorithm;
b. reduced preservation of subtle details in the Tomita and Tsuji case, but a marked improvement in the Nagao and Matsuyama case;

c. reduced immunity to shape distortion in the Tomita and Tsuji case, but enhanced immunity for the Nagao and Matsuyama algorithm; and

d. increased preservation of step edges with both algorithms.

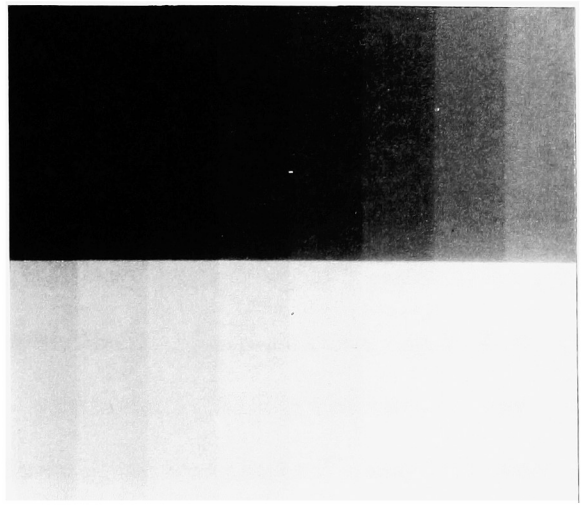
2. RECOMMENDATIONS

The first recommendation for further work in this area consists in varying the conditions of the t-test used for hypothesis testing (eg. vary the significance level α) in view of establishing a relationship between what is a statistically significant difference and a "practical" difference in performance.

Work could also be accomplished in determining an optimum cut-off contrast level for which the iterative Nagao and Matsuyama filter could effectively be used.

APPENDIX 1

INITIAL IMAGES PRIOR TO NOISE CORRUPTION



(a)

(b)

(c)

Original images of (a) gray scale, (b) girl's face, and (c) city center scene.

APPENDIX 2

GENERATION OF RANDOM NOISE

I. UNCORRELATED NOISE

Digital white noise can be artificially generated by creating a Gaussian distribution of gray levels with the use of a random number generator. The random generator provides a uniform distribution (i.e. equal probability of occurrence) for values between 0.0 (inclusive) and 1.0 (exclusive). In this study, a gaussian distribution was approximated by summing six uniformly distributed samples and properly scaling the sum, resulting in a zero mean and unit variance gaussian random variable of the form

$$n_i = K \sum_{i=1}^{6} (u_i - c) \qquad (1)$$

where: n_i is random noise normally distributed;

u_i is a random number with uniform distribution;

c shifts the mean to zero;

K scales the variance.

To get a zero mean distribution, the variable c is set to 0.499999995 for the REAL*4 type random numbers generated by the VAX computer used for this study (and not 0.50000000

since the random value 1.0 is exclusive).

The resulting noise will approximate a Gaussian distribution of the form

$$f(x) = \left|\frac{1}{b}\right| gaus\left|\frac{x}{b}\right| \qquad (2)$$

where b is a scaling factor. The gaus function of equation (2) can also be approximated by the convolution of six rectangle functions.

The value of the variable b may be estimated using the second moment method[23] as follows:

$$M_{2i} = \int_{-\infty}^{\infty} f(x)\ x^2\ dx \qquad (3)$$

Substituting (2) into (3), one gets:

$$M_{2i} = \int_{-\infty}^{\infty} \left(\frac{1}{b}\right) gaus\left(\frac{x}{b}\right) x^2\ dx \qquad (4)$$

And by representing the gaus function by its exponential expression:

$$M_{2i} = \int_{-\infty}^{\infty} \frac{1}{b} e^{-\pi (x/b)^2} x^2 \, dx \tag{5}$$

$$= \frac{b^2}{2\pi} \tag{6}$$

M_{2i} is also equal to the sum of the second moments of the six rectangles convolved to approximate the gaus function:

$$M_{2i} = M_{21} + M_{22} + \ldots + M_{26} \tag{7}$$

and for a rectangle function, the second moment is:

$$M_{21} = \int_{-\infty}^{\infty} f(x) \, x^2 \, dx = \int_{-1/2}^{1/2} 1 \, x^2 \, dx \tag{8}$$

$$= \frac{x^3}{3}\Big|_{-1/2}^{1/2} = \frac{1}{12} \tag{9}$$

Therefore, from (6) and (9):

$$M_{2i} = 6 \, M_{21} = \frac{1}{2} = \frac{b^2}{2\pi} \tag{10}$$

and

$$b = \sqrt{\pi} \tag{11}$$

Thus the gaus function is approximately

$$f(x) = \left(\frac{1}{\sqrt{\pi}}\right) gaus\left(\frac{x}{\sqrt{\pi}}\right) \tag{12}$$

and its variance is the second moment

$$\sigma^2 = M_{2i} = \frac{1}{2} \tag{13}$$

A normal distribution function of unit variance may be obtained by dividing the distribution by its standard deviation.[22] Therefore, to get unit variance gaussian noise distribution, the variable K in (1) will be set at the square root of 2 in the problem at hand. Therefore:

$$n_i = \sqrt{2} \sum_{i=1}^{6} (u_i - 0.49999995) \tag{14}$$

The variance or the standard deviation of the distribution can be adjusted if desired by simply multiplying the distribution by the square root of the desired variance or by the desired standard deviation respectively.

II. CORRELATED NOISE

To obtain the required correlated noise array, one starts with a 512 by 512 pixel array of uncorrelated noise created using the method described above. The convolution of a spread function across it (eg. the scanning with an aperture of finite size) will result in low-pass filtering of the uncorrelated noise. However, instead of convolving a spatial two-dimensional low-pass filter across the array, we will operate in the frequency domain for simplicity.

The uncorrelated noise array was transformed in the frequency domain using a two-dimensional Fourier transform. The resulting array was then multiplied point by point by a two-dimensional gaus function to reduce the high-frequency noise.

The gaus function used in this case is radially symmetrical and of the form:

$$Q(i,j) \; = \; gaus\left(\frac{i}{128}\right) gaus\left(\frac{j}{128}\right) \qquad (15)$$

where i and j are pixel coordinates. The scaling factor was set at 128 in this case in order to get a significant amount of correlation.

APPENDIX 3

COMPUTER PROGRAM: CREATION OF NOISE ARRAYS

```
*******************************************************************************
*                                                                             *
*       PROGRAM  IDENTIFICATION  :  NOISE.FOR                                 *
*                                                                             *
*       ------------------------------------                                  *
*                                                                             *
*       This program creates a 2-D noise array of up to 512 by 512 pixels     *
*       and adds the noise to 8 bit images.                                   *
*                                                                             *
*       Noise will be random with gaussian distribution, zero mean, and       *
*       unit variance.  The variance may be adjusted to a user-defined value  *
*       when the noise is subsequently added to an image.  A selection of     *
*       different types of noise will be available for addition to an image:  *
*                       uncorrelated noise                                    *
*                       correlated noise                                      *
*                       additive and/or multiplicative noise.                 *
*                                                                             *
*       A gaussian distribution is approximated by summing six uniformly      *
*       distributed samples ranging from 0.0 (inclusive) to 1.0 (exclusive)   *
*       (obtained from the random number generator), shifted to zero and      *
*       scaled to unit variance.                                              *
*                                                                             *
*                                                                             *
*       REQUIRED SUBROUTINES :   PIXIN.FOR                                    *
*                                FFT2D.FOR                                    *
*                                FFT2C     (LIB_IMSL SUBROUTINE ON VAXC)      *
*                                PIXOUT.FOR                                   *
*                                FRAM16.FOR                                   *
*                                                                             *
*******************************************************************************
*                                                                             *
*                                Marc R. Lapointe                             *
*                                                                             *
*                  M.S. Thesis,  Imaging  and  Photographic  Science          *
*                      Rochester  Institute  of  Technology                   *
*                                                                             *
*******************************************************************************
*                                                                             *
*       VARIABLE  IDENTIFICATION                                              *
*                                                                             *
*       ----------------------                                                *
*                                                                             *
*       IMAGE      Array with digitized picture values                        *
*       CIMAGE  =  Complex array with complex IMAGE data                      *
*       UNCNOISE=  Array of uncorrelated noise, zero mean, unit variance      *
*       CORNOISE=  Array of corrolated noise, zero mean, unit variance        *
*       NOISE      Array of noise of user-defined avg. and std.dev.           *
*                    ready for output to disk (with INTEGER*2 data).          *
*       CORRUPT    Array with image corrupted by noise                        *
*       N, M    =  Dimensions of IMAGE                                        *
*                                                                             *
*******************************************************************************
*                                                                             *
*       TYPE DECLARATION AND STORAGE ALLOCATION                               *
*                                                                             *
*       ---------------------------------------                               *
*                                                                             *
        INTEGER*2   IMAGE(512,512), NOISE(512,512), SEL1,SEL2,SEL3, AVG,STDEV
        INTEGER*2   CORRUPT(512,512), COUNT
        INTEGER*4   SEED
        REAL*4      UNCNOISE ( 512, 512 ), CORNOISE ( 512, 512 ), LOWER,UPPER
        REAL*4      AMEAN, ASTDEV
```

```
      COMPLEX*8   CIMAGE ( 512, 512 )
      CHARACTER*1 ANSWER
      PARAMETER   ( N   512, M   512 )
*                                                                       *
**********                                                  ************
*                                                                       *
*       COMPUTATION  BLOCK                                              *
*       ------------------                                              *
*                                                                       *
      COUNT   1

    1 WRITE ( 6, 1000 )
 1000 FORMAT( /, 1X, 'Do you want to :'. /, 5X,
     1 '1.  Create a noise file only ?', /, 5X,
     2 '2.  Add noise to an image as well ?', /, 5X,
     3 'Enter selection : ' $ )

      READ ( 5, * ) SEL1

      IF ( SEL1 .EQ. 1 ) GO TO 5


*     Read in original binary image                                     *

      CALL PIXIN ( IMAGE, N, M )     !Read in Original Image

*     This is a debug statement only to view image pixels.  Compile     *
*     program with:  FOR/D_LINES  option if you wish this statement     *
*     included in the program.                                          *

D     CALL FRAM16 ( IMAGE, N, M )    !View image pixels

*     Computation of uncorrelated gaussian noise array                  *
*                                                                       *
      ------------------------------------------------------------------

    5 WRITE ( 6, * ) ' '
      WRITE ( 6, * ) 'UNCORRELATED NOISE CASE'

      SEED = 7654321 + 1000 * COUNT

      DO  J   1, N

          DO  I = 1, M

              SUM   0.0

              DO  K = 1, 6

                VAR   RAN ( SEED )   0.499999995
                SUM   SUM + VAR

              END DO

            UNCNOISE ( I, J ) = SUM * SQRT ( 2.0 )

          END DO

      END DO


*     Producing an output file for display of uncorrelated noise alone  *
*                                                                       *
      ------------------------------------------------------------------

   10 WRITE ( 6, 1100 )
 1100 FORMAT(/,1X,'Do you want to create an output image for display',
     1 /, 1X, 'of uncorrelated noise alone ?   ( Y/N ) : ' $ )
      READ ( 5, '(A)', END   10 ) ANSWER
```

```
        IF ( ANSWER .EQ. 'N' .OR. ANSWER .EQ. 'n' ) GO TO 15

   11   WRITE ( 6, 1101 )
 1101   FORMAT ( /, 5X, 'Specify standard deviation of noise', /, 5X,
     1  'in gray level counts (i.e. INTEGER value)   : ' $ )
        READ ( 5, *, END   11 ) STDEV

   12   WRITE ( 6, 1102 )
 1102   FORMAT(/, 5X, 'Specify average gray level of noise (0-255) : '$)
        READ ( 5, *, END   12 ) AVG

*       Creating noise file according to user-defined paremeters                *

        DO  J   1, N

                DO  I   1, M

                  NOISE( I, J )= AVG + ININT( STDEV * UNCNOISE( I, J ))

                  IF ( NOISE ( I, J ) .LT. 0 ) THEN
                  NOISE ( I, J )   0
                  ENDIF

                  IF ( NOISE ( I, J ) .GT. 255 ) THEN
                  NOISE ( I, J ) - 255
                  ENDIF

                END DO

        END DO

*       Writing output file to disk                                             *

D       CALL FRAM16 ( NOISE, N, M )   !View image pixels

        CALL PIXOUT ( NOISE, N, M )   !Writing noise file to disk

   14   WRITE ( 6, 1110 )
 1110   FORMAT( /, 1X, ' Do you want to create another output image ',
     1  /,1X,' of noise alone with different parameters ? ( Y/N ) : '$ )
        READ ( 5, '(A)', END   14 ) ANSWER

        IF( ANSWER .EQ. 'Y' .OR. ANSWER .EQ. 'y' ) GO TO 11

   15   IF ( SEL1 .EQ. 1 ) GO TO 50


*       Addition of uncorrelated noise to original image                        *
*       ________________________________________________                        *

   16   WRITE ( 6, 1200 )
 1200   FORMAT(//,1X,'Type of uncorrelated noise to add to the image ?:'
     1  ,/,5X,'1. Additive only', /,5X, '2. Additive and multiplicative'
     2  ,/, 5X, '3. None (consider correlated noise only)', /, 5X,
     3  'Enter selection : ' $ )

        READ ( 5, * ) SEL2

        IF ( SEL2 .EQ. 3 ) GO TO 50
        IF ( SEL2 .EQ. 2 ) GO TO 20

*       Corrupting image with additive uncorrelated noise                       *

        WRITE ( 6, 1201 )
 1201   FORMAT( /, 5X, 'Specify standard deviation of noise', /,
     1  5X, 'in gray level counts (i.e. INTEGER value)  : ' $ )

        READ ( 5, * ) STDEV
```

```
        DO  J   1, N

                DO  I = 1, M

                  CORRUPT(I,J)=ININT( IMAGE(I,J) + UNCNOISE(I,J)*STDEV )

                  IF ( CORRUPT ( I, J ) .LT. 0 ) THEN
                  CORRUPT ( I, J ) = 0
                  ENDIF

                  IF ( CORRUPT ( I, J ) .GT. 255 ) THEN
                  CORRUPT ( I, J )   255
                  ENDIF

                END DO

        END DO

*       Writing corrupted image to disk                                      *

D       CALL  FRAM16 ( CORRUPT, N, M )    ! View image pixels

        CALL  PIXOUT ( CORRUPT, N, M )    ! Write noisy image to disk

        GO TO 30


*       Corrupting image with combined additive & multiplicative noise       *

   20   WRITE ( 6, 1300 )
 1300   FORMAT(/,1X,'To approximate combined additive and multiplicative
     1  noise, the'. /,1X,'uncorrelated noise values are scaled linearly
     2  between user-', /,  1X, 'defined values, and the result is added
     3  to the image'. /, 1X, 'Specify lower and upper limits for linear
     4  approximation : '$)

        READ ( 5, * ) LOWER, UPPER

        WRITE ( 6, 1301 )
 1301   FORMAT( /, 5X, 'Specify standard deviation of noise', /,
     1  5X, 'in gray level counts (i.e. INTEGER value)  : ' $ )

        READ ( 5, * ) STDEV

        DO  J   1, N

                DO  I = 1, M

                  SCALE   LOWER + ( UPPER-LOWER ) * ( IMAGE(I,J)/255 )

                  CORRUPT( I, J )   ININT( IMAGE( I, J ) +
     C                             UNCNOISE( I, J ) * STDEV * SCALE )

                  IF ( CORRUPT ( I, J ) .LT. 0 ) THEN
                  CORRUPT ( I, J )   0
                  ENDIF

                  IF ( CORRUPT ( I, J ) .GT. 255 ) THEN
                  CORRUPT ( I, J )   255
                  ENDIF

                END DO
        END DO

*       Writing corrupted image to disk                                      *

D       CALL  FRAM16 ( CORRUPT, N, M )    ! View image pixels
```

```
      CALL  PIXOUT ( CORRUPT, N, M )    ! Write noisy image to disk

   30 WRITE( 6, 1310 )
 1310 FORMAT( /, 1X, ' Do you want to add uncorrelated noise of ',
     1 /, 1X, ' different nature to this image ?  ( Y/N ) : ' $ )
      READ ( 5, '(A)', END   30 ) ANSWER

      IF ( ANSWER .EQ. 'Y' .OR. ANSWER .EQ. 'y' ) GO TO 16


*     Computation of correlated gaussian noise array                    *
*     ----------------------------------------------                    *

   50 WRITE ( 6, * ) ' '
      WRITE ( 6, * ) 'CORRELATED NOISE CASE'

*     Correlated noise is obtained by convolving a spread function      *
*     across the uncorrelated noise array.  For simplicity, this will   *
*     be done in the frequency domain by multiplying by the transform   *
*     of the spread function point by point.  Here we multiply by the   *
*     2-D gaus function:  gaus(I/128)*gaus(J/128).                      *
*                                                                       *
*     Center origin of uncorrelated noise array ( multiplying by        *
*     -1**(I+J) in the spatial domain centers origin in the frequency   *
*     domain.  Origin will be located at (257,257).                     *
*     Also convert array to complex type.                               *

      DO  J = 1, N

          DO  I   1, M

            CIMAGE(I,J)   CMPLX(  UNCNOISE(I,J)
     1                          * ( ( -1 ) ** ( I + J ) ) )

          END DO

      END DO


*     Take forward FFT of uncorrelated noise array                      *

      CALL  FFT2D ( CIMAGE, N, 1 )


*     Point by point multiplication by 2-D gaus function                *

      PI = 3.141592654

      DO  J   1, N

          DO  I   1, M

            GAUS2D= EXP( -PI * ( ( I   (N/2+1) ) / (N/4) ) ** 2 )
     C            * EXP( -PI * ( ( J   (N/2+1) ) / (N/4) ) ** 2 )

            CIMAGE( I, J )   CIMAGE( I, J ) * GAUS2D

          END DO

      END DO


*     Take reverse FFT of image, back to spatial domain                 *

      CALL  FFT2D ( CIMAGE, N, -1 )


*     Shift origin back to top left corner and convert back to real type  *
```

```
      DO  J   1, N
              DO  I   1, M
                CIMAGE( I, J )   CIMAGE( I, J ) * ( (-1) ** ( I + J ))
                CORNOISE( I, J )= REAL( CIMAGE( I, J ) )
              END DO
      END DO
*     Determination of the new mean and standard deviation of noise array  *
      SUM = 0.0
      DO  J   1, N
              DO  I   1, M
                SUM   SUM + CORNOISE( I, J )
              END DO
      END DO
      CMEAN   SUM / ( N * M )
      SUMSQ   0.0
      DO  J   1, N
              DO  I   1, M
                SUMSQ = SUMSQ + ( CORNOISE( I, J )   CMEAN ) ** 2
              END DO
      END DO
      CSTDEV = SQRT( SUMSQ / ( N * M   1 ) )

*     Producing an output file for display of correlated noise alone       *
*     ------------------------------------------------------------------   *
   60 WRITE ( 6, 1400 )
 1400 FORMAT(/,1X,'Do you want to create an output image for display',
     1 /, 1X, 'of correlated noise alone ?   ( Y/N ) : ' $ )
      READ ( 5, '(A)', END   60 ) ANSWER

      IF ( ANSWER .EQ. 'N' .OR. ANSWER .EQ. 'n' ) GO TO 65

   61 WRITE ( 6, 1401 )
 1401 FORMAT ( /, 5X, 'Specify standard deviation of noise', /, 5X,
     1 'in gray level counts (i.e. INTEGER value)   : ' $ )
      READ ( 5, *, END = 61 ) STDEV

   62 WRITE ( 6, 1402 )
 1402 FORMAT(/, 5X, 'Specify average gray level of noise (0-255) : '$)
      READ ( 5, *, END   62 ) AVG


*     Creating noise file according to user-defined parameters
      DO  J   1, N
              DO  I   1, M
```

```
                NOISE(I,J)=AVG + ININT(CORNOISE(I,J) * (STDEV/CSTDEV))

                IF ( NOISE( I, J ) .LT. 0 ) THEN
                NOISE ( I, J )   0
                END IF

                IF ( NOISE( I, J ) .GT. 255 ) THEN
                NOISE ( I, J )   255
                END IF

              END DO

        END DO

*       Writing output file to disk                                          *

D       CALL  FRAM16 ( NOISE, N, M )   !View image pixels

        CALL  PIXOUT ( NOISE, N, M )   !Writing noise file to disk

   63   WRITE ( 6, 1410 )
 1410   FORMAT( /, 1X, ' Do you want to create another output image of',
     1  /, 1X, ' noise alone with different parameters ? ( Y/N ) : ' $ )
        READ ( 5, '(A)', END   65 ) ANSWER

        IF ( ANSWER .EQ. 'Y' .OR. ANSWER .EQ. 'y' ) GO TO 61

   65   IF ( SEL1 .EQ. 1 ) GO TO 99999


*       Addition of correlated noise to original image                       *
*       ----------------------------------------------                       *

   68   WRITE ( 6, 1500 )
 1500   FORMAT( //,1X,'Type of correlated noise to add to the image ? :'
     1  ,/,5X, '1. Additive only', /,5X,'2. Additive and multiplicative'
     2  , /, 5X, '3. None  (Just get out of this program !!)', /, 5X,
     3  'Enter selection : ' $ )

        READ ( 5, * ) SEL3

        IF ( SEL3 .EQ. 3 ) GO TO 99999
        IF ( SEL3 .EQ. 2 ) GO TO 70


*       Corrupting image with additive correlated noise                      *

        WRITE ( 6, 1501 )
 1501   FORMAT( /, 5X, 'Specify standard deviation of noise', /,
     1  5X, 'in gray level counts (i.e. INTEGER value)  : ' $ )

        READ ( 5, * ) STDEV

        DO  J = 1, N

              DO  I   1, M

                CORRUPT( I, J )   ININT( IMAGE( I, J ) +
     C                             CORNOISE( I, J ) * STDEV / CSTDEV )

                IF ( CORRUPT( I, J ) .LT. 0 ) THEN
                CORRUPT( I, J )   0
                END IF

                IF ( CORRUPT( I, J ) .GT. 255 ) THEN
                CORRUPT( I, J ) = 255
                END IF
```

```
                END DO

        END DO

*       Writing corrupted image to disk                                 *

D       CALL  FRAM16 ( CORRUPT, N, M )    !View image pixels

        CALL  PIXOUT ( CORRUPT, N, M )    !Write image to disk

        GO TO 100


*       Corrupting image with combined additive and multiplicative noise *

   70   WRITE ( 6, 1600 )
 1600   FORMAT(/,1X,'To approximate combined additive and multiplicative
     1  noise, the'. /, 1X, 'correlated noise values are scaled linearly
     2  between user-', /, 1X,  'defined values, and the result is added
     3  to the image.', /, 1X,'Specify lower and upper limits for linear
     4  approximation : '$ )

        READ ( 5, * ) LOWER, UPPER

        WRITE ( 6, 1601 )
 1601   FORMAT( /, 5X, 'Specify standard deviation of noise', /,
     1  5X, 'in gray level counts (i.e. INTEGER value)  : ' $ )

        READ ( 5, * ) STDEV

        DO  J   1, N

                DO  I   1, M

                  SCALE = LOWER + ( UPPER-LOWER ) * ( IMAGE(I,J)/255 )

                  CORRUPT( I, J )   ININT( IMAGE( I, J ) +
     C                    CORNOISE( I, J ) * STDEV/CSTDEV * SCALE )

                  IF ( CORRUPT( I, J ) .LT. 0 ) THEN
                  CORRUPT( I, J )   0
                  END IF

                  IF ( CORRUPT( I, J ) .GT. 255 ) THEN
                  CORRUPT( I, J ) = 255
                  END IF

                END DO

        END DO

*       Writing corrupted image to disk                                 *

D       CALL  FRAM16 ( CORRUPT, N, M )  !View image pixels

        CALL  PIXOUT ( CORRUPT, N, M )    !Write image to disk

  100   WRITE ( 6, 1610 )
 1610   FORMAT( /, 1X, ' Do you want to add correlated noise of ',
     1  /, 1X, ' different nature to this image ?  ( Y/N ) : ' $ )
        READ ( 5, '(A)', END   100 ) ANSWER

        IF ( ANSWER .EQ. 'Y' .OR. ANSWER .EQ. 'y' ) GO TO 68

        COUNT = COUNT + 1

  110   WRITE ( 6, 1700 )
 1700   FORMAT( /, 1X, 'Do you want to run this progran again ',
```

```
      1  /, 1X, 'with another image ?  ( Y/N ) : ' $ )
         READ ( 5, '(A)', END   110 ) ANSWER

         IF ( ANSWER .EQ. 'Y' .OR. ANSWER .EQ. 'y' ) GO TO 1

99999    STOP

         END
```

```
*******************************************************************************
*                                                                             *
*       Program Identification : PIXIN.FOR                                    *
*                                                                             *
*       ---------------------------------                                     *
*                                                                             *
      SUBROUTINE PIXIN ( IMAGE , N , M )
*                                                                             *
*       This Sub-Program can be used to read in a direct access               *
*       disk file with binary, unformatted, LOGICAL*1 type data.              *
*       The data must be a 2-D array of numbers of size                       *
*       no greater than 512 X 512.  Smaller sizes may be used                 *
*       if necessary.  Similarly, the data must be quantized to               *
*       no greater than 8 bits   i.e. the gray levels of the image            *
*       must be within the range 0   255.                                     *
*       The LOGICAL*1 data is converted to INTEGER*2 type by                  *
*       the subroutine, and is returned as output to the calling              *
*       program.                                                              *
*       NOTE: The matrix read in is transposed by this subroutine.            *
*       This is done for better computational speed because of the            *
*       "column precedence" of Vax-11/780 FORTRAN compiler                    *
*                                                                             *
*********                                                            *********
*                                                                             *
*                     Nitin Sampat                                            *
*                                                                             *
*               M.S. Thesis Imaging and Photographic Sc.                      *
*                   Rochester Insitute of Technology.                         *
*                                                                             *
*********                                                            *********
*                                                                             *
*       Variable Identification :                                             *
*                                                                             *
*       -------------------------                                             *
*                                                                             *
*       IMAGE          2-D array with output INTEGER*2 data                   *
*       N              Lower bound of array image ( <   512 )                 *
*       M              Upper bound of array image ( < = 512 )                 *
*       LOGIC          1-D buffer array with LOGICAL*1 data                   *
*       FILENAME=      User supplied file to be read in and                   *
*                      converted to INTEGER*2 type data.                      *
*                                                                             *
*********                                                            *********
*                                                                             *
*       Type Declaration and Storage Allocation :                             *
*                                                                             *
*       ----------------------------------------                              *
*
      LOGICAL*1 LOGIC ( 512 )
      INTEGER*2 IMAGE ( N, M )
      CHARACTER*1    ANSWER, BELL
      CHARACTER*20    FILENAME
      PARAMETER ( BELL = CHAR ( 7 ) )
*                                                                             *
*********                                                            *********
*                                                                             *
*       Computation Block :                                                   *
*                                                                             *
*       ------------------                                                    *
*                                                                             *
*       Read in name of disk file with LOGICAL*1 type data :

5     WRITE ( 6, 100 )
100   FORMAT ( /, 1X, 'What is the name of the file with your',/,
     1           1X, 'image data (in binary form) :', $ )

      READ  ( 5, '(A)', ERR   1000, END = 5 ) FILENAME

*     Open direct access file at logical unit 1 :                             *

      OPEN ( 1, FILE   FILENAME, ACCESS   'DIRECT', STATUS = 'OLD',
     C       RECL   M / 4, FORM   'UNFORMATTED', ERR   1000 )
```

```
*       Read in LOGICAL*1 data from file at logical unit 1 and          *
*       convert this to INTEGER*2 type :                                *

        DO  J   1, M

        READ ( 1, REC   J, ERR   200 ) ( LOGIC ( K ), K   1, M )

                DO  I = 1, N

                IMAGE ( I, J )   LOGIC ( I )

*       The following IF statement is necessitated by the fact that     *
*       in one byte it is only possible to store numbers ranging        *
*       from -128 to + 127; this is because the leftmost bit is the     *
*       sign bit.  To store nos. upto 255, then, we assume that a       *
*       factor of 256 was subtracted while writing to LOGICAL*1 data    *
*       type.  As such, we now have to add this factor while converting *
*       the data to INTEGER*2 type.                                     *

                  IF ( IMAGE ( I, J ) .LT. 0 )
     C            IMAGE ( I, J )   IMAGE ( I, J ) + 256

           END DO

        END DO

*       Close direct access file at logical unit 1 :                    *

        CLOSE ( UNIT = 1, STATUS   'KEEP' )

*       Return to calling program :                                     *

        WRITE ( 6, * ) ' '
        WRITE ( 6, * ) 'LOGICAL*1 data read in from file: ', FILENAME
        WRITE ( 6, * ) 'and converted to INTEGER*2 type'
        WRITE ( 6, * ) '...returning to calling program'

        RETURN

*       Error message block :                                           *
*       ____________________                                            *

1000    WRITE ( 6, * ) BELL
        WRITE ( 6, * ) ' '
        WRITE ( 6, * )'You have encountered an error while specifying'
        WRITE ( 6, * )'your image data file. Please check the following:'
        WRITE ( 6, * ) ' '
        WRITE ( 6, * )'1. Did you enter the correct file name ? '
        WRITE ( 6, * )'2. Do you have the required LOGICAL*1 image'
        WRITE ( 6, * )'   data file in your directory ?'
1050    WRITE ( 6, 2000 )
2000    FORMAT(/,1X,'Would you like to try another file name (Y/N) :',$ )

        READ  ( 5, '(A)', END = 1050 ) ANSWER

        IF ( ANSWER .EQ. 'Y' .OR. ANSWER .EQ. 'y' ) THEN

        GOTO 5                  !Return for correct file name

        ELSE

        WRITE ( 6, 3000 )
3000    FORMAT (/,1X,'Aborting program ....' )

        END IF

200     WRITE ( 6, * ) BELL
        WRITE ( 6, 4000) FILENAME
```

```
4000    FORMAT(/,1X, 'Error in reading from file :', A, /,
        1         1X, 'Note : Data in file should be in binary form',
        1         1X, '       stored as Logical*1 type in a direct'
        1         1X, '       file' , /)
        STOP 'Aborting program'

        END
```

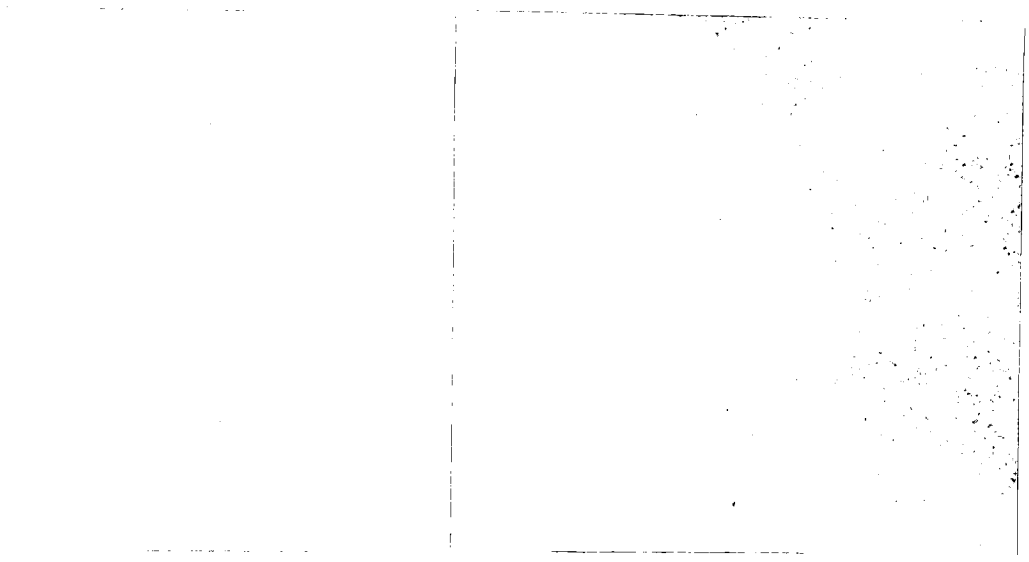
```
C*************************************************************************
C*                                                                       *
C*       Program Identification : PIXOUT.FOR                             *
C*       ___________________________________                             *
C*                                                                       *
      SUBROUTINE PIXOUT ( IMAGE , N , M )
C*                                                                       *
C*       This Sub-Program can be used to write to a direct access        *
C*       disk file, binary, unformatted, LOGICAL*1 type data.            *
C*       The input data must be a 2-D array of numbers (INTEGER*2 type)  *
C*       of size no greater than 512 X 512.  Smaller sizes may be used   *
C*       if necessary.  Similarly, the data must be quantized to         *
C*       no greater than 8 bits   i.e. the gray levels of the image      *
C*       must be within the range 0   255.                               *
C*       The INTEGER*2 data is converted to LOGICAL*1 type by            *
C*       this subroutine, and is written to user specified output        *
C*       file.                                                           *
C*       NOTE: The matrix written to disk is transposed as a result      *
C*       of using this subroutine. This is done for better computational *
C*       efficiency obtained by taking advantage of the "column          *
C*       precedence" of the VAX-11/780 Fortran compiler                  *
C*                                                                       *
C*********                                                     *********
C*                                                                       *
C*                     Nitin Sampat                                      *
C*                                                                       *
C*               M.S. thesis Imaging and Photographic Sc.                *
C*                  Rochester Institute of Technology.                   *
C*                                                                       *
C*********                                                     *********
C*                                                                       *
C*       Variable Identification :                                       *
C*       _________________________                                       *
C*                                                                       *
C*       IMAGE          2-D array with input INTEGER*2 data              *
C*       N              Lower bound of array image ( <   512 )           *
C*       M              Upper bound of array image ( <   512 )           *
C*       LOGIC          1-D buffer with LOGICAL*1 data                   *
C*       FILENAME=      User supplied file to be written to after        *
C*                      data is converted type.                          *
C*                                                                       *
C*********                                                     *********
C*                                                                       *
C*       Type Declaration and Storage Allocation :                       *
C*       _________________________________                               *
C*
      LOGICAL*1 LOGIC ( 512 )
      INTEGER*2 IMAGE ( N, M )
      CHARACTER*20    FILENAME
      CHARACTER*1    ANSWER, BELL
      PARAMETER ( BELL = CHAR ( 7 ) )
C*                                                                       *
C*********                                                     *********
C*                                                                       *
C*       Computation Block :                                             *
C*       ___________________                                             *
C*                                                                       *
C*       Read in name of disk file with LOGICAL*1 type data :            *
C*       to be written :
C
5     WRITE ( 6, 100 )
100   FORMAT(/, 1X,'Enter a name for the file in which you wish',/,
     1 1X,'to store your image data', /,
     1 1X'(data will be stored in binary,unformatted form) :' $ )

      READ  ( 5, '(A)', END   5, ERR   1000 ) FILENAME
                                      !Read in name of file
```

```
*       Open a direct access file at Logical Unit 2 :                    *

        OPEN ( 2, FILE = FILENAME, ACCESS   'DIRECT', STATUS = 'NEW',
     C         RECL   N/4, FORM   'UNFORMATTED', ERR   1000 )

*       Convert INTEGER*2 data to LOGICAL*1 type and write to file at    *
*       Logical Unit 2 :                                                 *

        DO  J = 1, N

                DO  I   1, M

*       The following IF statement is necessitated by the fact that we   *
*       can only write numbers ranging from -128 to +127 in 1 byte.      *
*       This is because the leftmost biit in a byte is the sign bit.     *
*       As such, to write numbers ranging from 0   255 (in case the      *
*       image is quantized to 8 bits/pixel), we have to subtract a       *
*       factor of 256 from the data before converting INTEGER*2 to       *
*       LOGICAL*1 type data.  Ofcourse, this means that when reading     *
*       from a file containing this data we'll have to add the same      *
*       factor.                                                          *

                IF ( IMAGE ( I, J ) .GT. 127 )
        1       IMAGE ( I, J )   IMAGE ( I, J )   256

                LOGIC ( I ) = IMAGE ( I, J )      !Conversion to Logical*1

                END DO

           WRITE ( 2, REC   J, ERR   3000 ) ( LOGIC ( K ), K   1, M )

        END DO

*       Close file at Logical Unit 2 :                                   *

        CLOSE ( UNIT   2 , STATUS   'KEEP' )

*       Return to calling program :                                      *

        WRITE ( 6, * ) ' '
        WRITE ( 6, * ) 'Integer*2 data converted to logical*1 type'
        WRITE ( 6, * ) 'and stored in file : ', FILENAME
        WRITE ( 6, * ) ' '
        WRITE ( 6, * ) 'Now returning back to calling program..'

        RETURN

*       Error message block :                                            *

1000    WRITE ( 6, * ) BELL
        WRITE ( 6, * ) ' '
        WRITE ( 6, * ) 'You have encountered an error while specifying'
        WRITE ( 6, * ) 'your filename. Please check the following :'
        WRITE ( 6, * ) ' '
        WRITE ( 6, * ) '1.  Did you type in the correct file name ?'
        WRITE ( 6, * ) '2.  Do you have enough memory in your disk quota?'
1050    WRITE ( 6, 2000 )
2000    FORMAT (/,1X,'Would you like to try another file name ? (Y/N) :'$)

        READ  ( 5, '(A)', END   1050 ) ANSWER

        IF ( ANSWER .EQ. 'Y' .OR. ANSWER .EQ. 'y' ) THEN

        GOTO 5                    !Return for correct filename

        ELSE

        STOP 'Aborting program....'
```

```
      END IF

3000  WRITE ( 6, 4000 ) FILENAME
4000  FORMAT(/, 1X, 'Error while writing to file :', A, / ,
      1         1X, 'You may have exceeded your disk quota !' )

      STOP 'Aborting program.....'

      END
```

APPENDIX 4

NOISE ARRAYS



(a) (b)

Arrays of (a) uncorrelated and (b) correlated noise

APPENDIX 5

SAMPLE OF NOISE CORRUPTED IMAGES

Images corrupted with high level
of correlated noise

APPENDIX 6

COMPUTER PROGRAMS: NOISE FILTERING

```
************************************************************************************
*                                                                                  *
*       PROGRAM  IDENTIFICATION  :  TOMITA.FOR                                     *
*                                                                                  *
*       ---------------------------------------                                    *
*                                                                                  *
*       This program performs local noise filtering on 8 bit images, up           *
*       to 512 by 512 pixels.                                                      *
*                                                                                  *
*       One algorithm is used in this program:                                     *
*                 Tomita and Tsuji                                                 *
*                                                                                  *
*       REQUIRED SUBROUTINES :  PIXIN.FOR                                          *
*                               PIXOUT.FOR                                         *
*                                                                                  *
************************************************************************************
*                                                                                  *
*                               Marc R. Lapointe                                   *
*                M.S. Thesis,  Imaging  and  Photographic  Science                 *
*                      Rochester Institute of Technology                           *
*                                                                                  *
************************************************************************************
*                                                                                  *
*       VARIABLE  IDENTIFICATION                                                   *
*       ------------------------                                                   *
*                                                                                  *
*       IM       =  Array with digitized picture values                            *
*       TMP, AR  =  Temporary working arrays                                       *
*       OUT      =  Output array with filtered image                               *
*       AVG(n)   =  Average image gray level in the nth neighborhood               *
*       VAR(n)      Variance in image gray levels in the nth neighborhood          *
*       RANGE(n)=   Range of gray levels in the nth neighborhood                   *
*                                                                                  *
************************************************************************************
*                                                                                  *
*       TYPE DECLARATION AND STORAGE ALLOCATION                                    *
*       ---------------------------------------                                    *
*                                                                                  *
*                                                                                  *
      INTEGER*2   IM(512,512), OUT(512,512), AR(512,512)
      INTEGER*2   REGION, MINI, MAXI
      INTEGER*4   TMP(512,512), SUM, SUMSQ
      REAL*4      AVG( 9 ), VAR( 9 ), RANGE( 9 ), RANLO
      PARAMETER   ( N = 512, M   512 )
*                                                                                  *
*********                                                            ***********
*                                                                                  *
*       COMPUTATION  BLOCK                                                         *
*       ------------------                                                         *
*                                                                                  *
*
*       Read in the image to be filtered                                           *
*       --------------------------------                                           *
*
  1   WRITE ( 6, * ) ' '
      WRITE ( 6, * ) ' INPUT OF IMAGE TO BE FILTERED'

      CALL  PIXIN( IM, N, M )
```

```
*-----------------------------------------------------------------------
*
*       Run original Tomita and Tsuji algorithm
*
        ---------------------------------------

        WRITE ( 6, * ) ' '
        WRITE ( 6, * ) ' Now filtering the image with the '
        WRITE ( 6, * ) ' original Tomita and Tsuji algorithm.'
        WRITE ( 6, * ) '      Please be patient.  Running...'

*       Transfer original image to a temporary working array to be
*       used by the algorithm.  (This is to allow squaring of image
*       gray level values without integer overflow by using INTEGER*4
*       type variable vice INTEGER*2 ).

        DO  J = 1, N
              DO  I   1, M
              TMP( I, J ) = IM( I, J )
              END DO
        END DO

        DO  J   1, N              ! Start scanning of image
              DO  I   1, M        ! pixel by pixel


*       Compute average and variance of pixels in first neighborhood
*       ( upper right region )

   10   IF( I .LE. 2 .OR. J .GT. ( N- 2 ) ) THEN
        VAR( 1 )   65536.0        ! If neighborhood outside  image edges,
        GO TO 20                ! then do not compute AVG(1) and VAR(1)
        END IF                    ! and go to next neighborhood.

*       Compute AVG(1) and VAR(1)

        SUM   = TMP(I-2,J  )    + TMP(I-2,J+1)    + TMP(I-2,J+2)    +
     1          TMP(I-1,J  )    + TMP(I-1,J+1)    + TMP(I-1,J+2)    +
     2          TMP(I  ,J  )    + TMP(I  ,J+1)    + TMP(I  ,J+2)

        SUMSQ   TMP(I-2,J  )**2 + TMP(I-2,J+1)**2 + TMP(I-2,J+2)**2 +
     1          TMP(I-1,J  )**2 + TMP(I-1,J+1)**2 + TMP(I-1,J+2)**2 +
     2          TMP(I  ,J  )**2 + TMP(I  ,J+1)**2 + TMP(I  ,J+2)**2

        AVG( 1 )   SUM / 9

        VAR( 1 )   ( ( 9 * SUMSQ )   ( SUM ** 2 ) ) / 72.0


*       Compute average and variance of pixels in second neighborhood
*       ( lower right region )

   20   IF( I .GT. ( M   2 ) .OR. J .GT. ( N   2 ) ) THEN
        VAR( 2 ) = 65536.0
        GO TO 30
        END IF

*       Compute AVG(2) and VAR(2)

        SUM   = TMP(I  ,J  )    + TMP(I  ,J+1)    + TMP(I  ,J+2)    +
     1          TMP(I+1,J  )    + TMP(I+1,J+1)    + TMP(I+1,J+2)    +
     2          TMP(I+2,J  )    + TMP(I+2,J+1)    + TMP(I+2,J+2)

        SUMSQ = TMP(I  ,J  )**2 + TMP(I  ,J+1)**2 + TMP(I  ,J+2)**2 +
     1          TMP(I+1,J  )**2 + TMP(I+1,J+1)**2 + TMP(I+1,J+2)**2 +
     2          TMP(I+2,J  )**2 + TMP(I+2,J+1)**2 + TMP(I+2,J+2)**2

        AVG( 2 ) = SUM / 9
```

```
      VAR( 2 ) = ( ( 9 * SUMSQ )   ( SUM ** 2 ) ) / 72.0

*     Compute average and variance of pixels in third neighborhood
*     ( lower left region )

30    IF( I .GT. ( M   2 ) .OR. J .LE. 2 ) THEN
      VAR( 3 ) = 65536.0
      GO TO 40
      END IF

*     Compute AVG(3) and VAR(3)

      SUM     TMP(I  ,J-2)    + TMP(I  ,J-1)    + TMP(I  ,J  )    +
     1        TMP(I+1,J-2)    + TMP(I+1,J-1)    + TMP(I+1,J  )    +
     2        TMP(I+2,J-2)    + TMP(I+2,J-1)    + TMP(I+2,J  )

      SUMSQ = TMP(I  ,J-2)**2 + TMP(I  ,J-1)**2 + TMP(I  ,J  )**2 +
     1        TMP(I+1,J-2)**2 + TMP(I+1,J-1)**2 + TMP(I+1,J  )**2 +
     2        TMP(I+2,J-2)**2 + TMP(I+2,J-1)**2 + TMP(I+2,J  )**2

      AVG( 3 )   SUM / 9

      VAR( 3 )   ( ( 9 * SUMSQ )   ( SUM ** 2 ) ) / 72.0


*     Compute average and variance of pixels in forth neighborhood
*     ( upper left region )

40    IF( I .LE. 2 .OR. J .LE. 2 ) THEN
      VAR( 4 )   65536.0
      GO TO 50
      END IF

*     Compute AVG(4) and VAR(4)

      SUM     TMP(I-2,J-2)    + TMP(I-2,J-1)    + TMP(I-2,J  )    +
     1        TMP(I-1,J-2)    + TMP(I-1,J-1)    + TMP(I-1,J  )    +
     2        TMP(I  ,J-2)    + TMP(I  ,J-1)    + TMP(I  ,J  )

      SUMSQ = TMP(I-2,J-2)**2 + TMP(I-2,J-1)**2 + TMP(I-2,J  )**2 +
     1        TMP(I-1,J-2)**2 + TMP(I-1,J-1)**2 + TMP(I-1,J  )**2 +
     2        TMP(I  ,J-2)**2 + TMP(I  ,J-1)**2 + TMP(I  ,J  )**2

      AVG( 4 )   SUM / 9

      VAR( 4 ) = ( ( 9 * SUMSQ )   ( SUM ** 2 ) ) / 72.0


*     Compute average and variance of pixels in fifth neighborhood
*     ( central region )

50    IF( I.EQ.1 .OR. I.EQ.M .OR. J.EQ.1 .OR. J.EQ.N ) THEN
      VAR( 5 ) = 65536.0
      GO TO 60
      END IF

*     Compute AVG(5) and VAR(5)

      SUM     TMP(I-1,J-1)    + TMP(I-1,J  )    + TMP(I-1,J+1)    +
     1        TMP(I  ,J-1)    + TMP(I  ,J  )    + TMP(I  ,J+1)    +
     2        TMP(I+1,J-1)    + TMP(I+1,J  )    + TMP(I+1,J+1)

      AVG( 5 )   SUM / 9

      VAR( 5 )   ( ( 9 * SUMSQ )   ( SUM ** 2 ) ) / 72.0
```

```
*       Determine which neighborhood has lowest variance

   60   VARLO = 65536.0
        DO  K = 1, 5
                IF( VAR( K ) .LE. VARLO ) THEN
                VARLO  = VAR( K )
                REGION   K
                END IF
        END DO

*       Assing average gray level of corresponding neighborhood to
*       pixel of interest.

        AVE   AVG( REGION )
        OUT( I, J )   ININT( AVE )


                END DO                  ! End of Tomita and Tsuji's
        END DO                  ! algorithm

        WRITE ( 6, * ) '      Completed!'

*       Write filtered image to disk

        CALL  PIXOUT( OUT, N, M )       ! Write image to disk

99999   STOP

        END
```

```
*******************************************************************************
*                                                                             *
*       PROGRAM  IDENTIFICATION  :  TOMLAP.FOR                                *
*                ____________________________                                 *
*                                                                             *
*       This program performs local noise filtering on 8 bit images, up      *
*       to 512 by 512 pixels.                                                 *
*                                                                             *
*       One algorithm is used in this program:                                *
*                 Modified Tomita and Tsuji                                   *
*                                                                             *
*       In the modified versions of both algorithms, the calculation of       *
*       the range is substituted to that of the variance in each of the       *
*       neighborhoods surrounding the pixel of interest.                      *
*                                                                             *
*                                                                             *
*       REQUIRED SUBROUTINES :  PIXIN.FOR                                     *
*                               PIXOUT.FOR                                    *
*                                                                             *
*******************************************************************************
*                                                                             *
*                               Marc R. Lapointe                              *
*                M.S. Thesis,  Imaging  and  Photographic  Science            *
*                      Rochester Institute of Technology                      *
*                                                                             *
*******************************************************************************
*                                                                             *
*       VARIABLE  IDENTIFICATION                                              *
*                 ______________________                                      *
*                                                                             *
*       IM          Array with digitized picture values                       *
*       TMP, AR     Temporary working arrays                                  *
*       OUT         Output array with filtered image                          *
*       AVG(n)      Average image gray level in the nth neighborhood          *
*       VAR(n)      Variance in image gray levels in the nth neighborhood     *
*       RANGE(n)=   Range of gray levels in the nth neighborhood              *
*                                                                             *
*******************************************************************************
*                                                                             *
*       TYPE DECLARATION AND STORAGE ALLOCATION                               *
*       _______________________________________                               *
*                                                                             *
      INTEGER*2   IM(512,512), OUT(512,512), AR(512,512)
      INTEGER*2   MIN(2,512), MAX(2,512), REGION, MINI, MAXI, RANLO
      INTEGER*4   SUM
      REAL*4      AVE
      PARAMETER   ( N = 512, M   512 )
*                                                                             *
*********                                                        ************
*                                                                             *
*       COMPUTATION  BLOCK                                                    *
*       __________________                                                    *
*                                                                             *

*       Read in the image to be filtered
*       ________________________________

    1 WRITE ( 6, * ) ' '
      WRITE ( 6, * ) ' INPUT OF IMAGE TO BE FILTERED'

      CALL  PIXIN( IM, N, M )

*-------------------------------------------------------------------------------
*
*       Run modified Tomita and Tsuji algorithm
*       _______________________________________

  100 WRITE ( 6, * ) ' '
```

```
WRITE ( 6, * ) ' Now filtering the image with the '
WRITE ( 6, * ) ' modified Tomita and Tsuji algorithm.'
WRITE ( 6, * ) '      Please be patient.  Running...'

DO  I = 2, M-2, 2
  DO  J   1, N

  IF ( IM(I+1,J) .LT. IM(I,J) ) THEN
    MIN(1,J) = IM(I+1,J)
    MIN(2,J)   IM(I+1,J)
    MAX(1,J)   IM(I,J)
    MAX(2,J)   IM(I,J)
  ELSE
    MIN(1,J)   IM(I,J)
    MIN(2,J)   IM(I,J)
    MAX(1,J) = IM(I+1,J)
    MAX(2,J) = IM(I+1,J)
  END IF

  IF ( IM(I-1,J) .LT. MIN(1,J) ) THEN
    MIN(1,J)   IM(I-1,J)
  ELSE IF ( IM(I-1,J) .GT. MAX(1,J) ) THEN
    MAX(1,J)   IM(I-1,J)
  END IF

  IF ( IM(I+2,J) .LT. MIN(2,J) ) THEN
    MIN(2,J) = IM(I+2,J)
  ELSE IF ( IM(I+2,J) .GT. MAX(2,J) ) THEN
    MAX(2,J)   IM(I+2,J)
  END IF

  END DO

  DO  J   2, N-1

    MINI = MIN(I,J)
    MAXI = MAX(I,J)

    IF ( MIN(1,J-1) .LT. MINI ) THEN
      MINI   MIN(1,J-1)
    ELSE IF ( MAX(1,J-1) .GT. MAXI ) THEN
      MAXI   MAX(1,J-1)
    END IF

    IF ( MIN(1,J+1) .LT. MINI ) THEN
      MINI   MIN(1,J+1)
    ELSE IF ( MAX(1,J+1) .GT. MAXI ) THEN
      MAXI = MAX(1,J+1)
    END IF

   AR(I,J) = MAXI   MINI

    MINI   MIN(2,J)
    MAXI   MAX(2,J)

    IF ( MIN(2,J-1) .LT. MINI ) THEN
      MINI   MIN(2,J-1)
    ELSE IF ( MAX(2,J-1) .GT. MAXI ) THEN
      MAXI = MAX(2,J-1)
    END IF

    IF ( MIN(2,J+1) .LT. MINI ) THEN
      MINI = MIN(2,J+1)
    ELSE IF ( MAX(2,J+1) .GT. MAXI ) THEN
      MAXI   MAX(2,J+1)
    END IF

   AR(I+1,J) = MAXI   MINI
```

```
      END DO
    END DO

    DO  I   1, M
      AR(1,I) = 256
      AR(M,I) = 256
      AR(I,1) = 256
      AR(I,M) = 256
    END DO

*   Determine which neighborhood has lowest range for each pixel and
*   assign averege gray level of that neighborhood to the output image.

    DO  I   2, M-1
      DO  J   2, N-1

      RANLO = AR(I,J)
      REGION= 5

      IF ( AR(I-1,J+1) .LT. RANLO ) THEN
        RANLO = AR(I-1,J+1)
        REGION= 1
      END IF

      IF ( AR(I+1,J+1) .LT. RANLO ) THEN
        RANLO = AR(I+1,J+1)
        REGION= 2
      END IF

      IF ( AR(I+1,J-1) .LT. RANLO ) THEN
        RANLO   AR(I+1,J-1)
        REGION= 3
      END IF

      IF ( AR(I-1,J-1) .LT. RANLO ) THEN
        RANLO = AR(I-1,J-1)
        REGION= 4
      END IF

*   Assign average gray level of corresponding neighborhood to
*   pixel of interest.

      SUM   0

        DO  JJ = 1, 3
            DO  II   1, 3

            IF( REGION .EQ. 1 ) THEN
              SUM   SUM + IM( I + II   3, J + JJ   1 )
            ELSEIF( REGION .EQ. 2 ) THEN
              SUM   SUM + IM( I + II   1, J + JJ   1 )
            ELSEIF( REGION .EQ. 3 ) THEN
              SUM = SUM + IM( I + II   1, J + JJ   3 )
            ELSEIF( REGION .EQ. 4 ) THEN
              SUM   SUM + IM( I + II   3, J + JJ   3 )
            ELSEIF( REGION .EQ. 5 ) THEN
              SUM   SUM + IM( I + II   2, J + JJ   2 )
            END IF

            END DO
        END DO

        AVE   SUM / 9
        OUT( I, J )   ININT( AVE )

      END DO                    ! End of modified Tomita and
    END DO                      ! Tsuji algorithm.
```

```
      WRITE ( 6, * ) '      Completed!'
*     Write filtered image to disk
      CALL  PIXOUT( OUT, N, M )        ! Write image to disk
99999 STOP
      END
```

```
*******************************************************************************
*                                                                             *
*       PROGRAM  IDENTIFICATION  :  NAGAO.FOR                                 *
*                ____________________________                                 *
*                                                                             *
*       This program performs local noise filtering on 8 bit images, up      *
*       to 512 by 512 pixels.                                                 *
*                                                                             *
*       One algorithm is used in this program:                                *
*                Nagao and Matsuyama                                          *
*                                                                             *
*       In the modified versions of both algorithms, the calculation of       *
*       the range is substituted to that of the variance in each of the       *
*       neighborhoods surrounding the pixel of interest.                      *
*                                                                             *
*                                                                             *
*       REQUIRED SUBROUTINES :  PIXIN.FOR                                     *
*                               PIXOUT.FOR                                    *
*                                                                             *
*******************************************************************************
*                                                                             *
*                                Marc R. Lapointe                             *
*                M.S. Thesis,  Imaging  and  Photographic  Science            *
*                      Rochester Institute of Technology                      *
*                                                                             *
*******************************************************************************
*                                                                             *
*       VARIABLE  IDENTIFICATION                                              *
*                 ______________________                                      *
*                                                                             *
*       IM      =  Array with digitized picture values                        *
*       TMP, AR =  Temporary working arrays                                   *
*       OUT     =  Output array with filtered image                           *
*       AVG(n)     Average image gray level in the nth neighborhood           *
*       VAR(n)  =  Variance in image gray levels in the nth neighborhood      *
*       RANGE(n)=  Range of gray levels in the nth neighborhood               *
*                                                                             *
*******************************************************************************
*                                                                             *
*       TYPE DECLARATION AND STORAGE ALLOCATION                               *
*       _______________________________________                               *
*                                                                             *
      INTEGER*2   IM(512,512), OUT(512,512), AR(512,512)
      INTEGER*2   REGION, MINI, MAXI
      INTEGER*4   TMP(512,512), SUM, SUMSQ
      REAL*4      AVG( 9 ), VAR( 9 ), RANGE( 9 ), RANLO
      PARAMETER   ( N   512, M   512 )
*                                                                             *
*********                                                      **************
*                                                                             *
*       COMPUTATION  BLOCK                                                    *
*       __________________                                                    *
*                                                                             *

*       Read in the image to be filtered
*       ________________________________

    1 WRITE ( 6, * ) ' '
      WRITE ( 6, * ) ' INPUT OF IMAGE TO BE FILTERED'

      CALL  PIXIN( IM, N, M )

*-------------------------------------------------------------------------------
*
*       Run original Nagao and Matsuyama algorithm
*       __________________________________________

  200 WRITE ( 6, * ) ' '
```

```
      WRITE ( 6, * ) ' Now filtering the image with the '
      WRITE ( 6, * ) ' original Nagao and Matsuyama algorithm.'
      WRITE ( 6, * ) '      Please be patient.  Running...'

*     Transfer original image to a temporary working array to be
*     used by the algorithm.  ( This is to preserve the original
*     image intact, as the algorithm is iterative and will change
*     the image from one pass to the next, and to allow squaring
*     of gray level values by using INTEGER*4 type variable vice
*     INTEGER*2.

      DO  J = 1, N
            DO  I   1, M
            TMP( I, J )   IM( I, J )
            END DO
      END DO

*     Begin iterative process

      DO  L   1, 5              ! Iterate filter 5 times

*     Perform filtering on transferred image

      DO  J   1, N              ! Start scanning of image
            DO  I = 1, M        ! pixel by pixel.

*     Compute average and variance of pixels in first neighborhood
*     ( upper right region )

  210 IF( I .LE. 2 .OR. J .GT. ( N   2 ) ) THEN
      VAR( 1 )   65536.0        ! If neighborhood outside image edges,
      GO TO 220                 ! then do not compute AVG(1) and VAR(1)
      END IF                    ! and go to next neighborhood.

*     Compute AVG(1) and VAR(1)

      SUM   =                     TMP(I-2,J+1)    + TMP(I-2,J+2)    +
     1          TMP(I-1,J  )    + TMP(I-1,J+1)    + TMP(I-1,J+2)    +
     2          TMP(I  ,J  )    + TMP(I  ,J+1)

      SUMSQ                       TMP(I-2,J+1)**2 + TMP(I-2,J+2)**2 +
     1          TMP(I-1,J  )**2 + TMP(I-1,J+1)**2 + TMP(I-1,J+2)**2 +
     2          TMP(I  ,J  )**2 + TMP(I  ,J+1)**2

      AVG( 1 ) = SUM / 7

      VAR( 1 )   ( ( 7 * SUMSQ )   ( SUM ** 2 ) ) / 42

*     Compute average and variance of pixels in second neighborhood
*     ( center right region )

  220 IF( I .EQ. 1 .OR. I .EQ. M .OR. J .GT. ( N   2 ) ) THEN
      VAR( 2 )   65536.0
      GO TO 230
      END IF

*     Compute AVG(2) and VAR(2)

      SUM                         TMP(I-1,J+1)    + TMP(I-1,J+2)    +
     1          TMP(I  ,J  )    + TMP(I  ,J+1)    + TMP(I  ,J+2)    +
     2                          TMP(I+1,J+1)    + TMP(I+1,J+2)

      SUMSQ                       TMP(I-1,J+1)**2 + TMP(I-1,J+2)**2 +
     1          TMP(I  ,J  )**2 + TMP(I  ,J+1)**2 + TMP(I  ,J+2)**2 +
     2                          TMP(I+1,J+1)**2 + TMP(I+1,J+2)**2

      AVG( 2 )   SUM / 7
```

```
        VAR( 2 ) = ( ( 7 * SUMSQ )   ( SUM ** 2 ) ) / 42

*       Compute average and variance of pixels in third neighborhood
*       ( lower right region )

  230   IF( I .GT. ( M   2 ) .OR. J .GT. ( N   2 ) ) THEN
        VAR( 3 ) = 65536.0
        GO TO 240
        END IF

*       Compute AVG(3) and VAR(3)

        SUM     TMP(I  ,J  )    + TMP(I  ,J+1)    +
     1          TMP(I+1,J  )    + TMP(I+1,J+1)    + TMP(I+1,J+2) +
     2                            TMP(I+2,J+1)    + TMP(I+2,J+2)

        SUMSQ   TMP(I  ,J  )**2 + TMP(I  ,J+1)**2 +
     1          TMP(I+1,J  )**2 + TMP(I+1,J+1)**2 + TMP(I+1,J+2)**2 +
     2                            TMP(I+1,J+1)**2 + TMP(I+1,J+2)**2

        AVG( 3 )   SUM / 7

        VAR( 3 )   ( ( 7 * SUMSQ )   ( SUM ** 2 ) ) / 42

*       Compute average and variance of pixels in forth neighborhood
*       ( lower center region )

  240   IF( I .GT. ( M   2 ) .OR. J .EQ. 1 .OR. J .EQ. N ) THEN
        VAR( 4 )   65536.0
        GO TO 250
        END IF

*       Compute AVG(4) and VAR(4)

        SUM   =                   TMP(I  ,J  )    +
     1          TMP(I+1,J-1)    + TMP(I+1,J  )    + TMP(I+1,J+1)    +
     2          TMP(I+2,J-1)    + TMP(I+2,J  )    + TMP(I+2,J+1)

        SUMSQ =                     TMP(I  ,J  )**2 +
     1          TMP(I+1,J-1)**2 + TMP(I+1,J  )**2 + TMP(I+1,J+1)**2 +
     2          TMP(I+2,J-1)**2 + TMP(I+2,J  )**2 + TMP(I+2,J+1)**2

        AVG( 4 )   SUM / 7

        VAR( 4 )   ( ( 7 * SUMSQ )   ( SUM ** 2 ) ) / 42

*       Compute average and variance of pixels in fifth neighborhood
*       ( lower left region )

  250   IF( I .GT. ( M   2 ) .OR. J .LE. 2 ) THEN
        VAR( 5 )   65536.0
        GO TO 260
        END IF

*       Compute AVG(5) and VAR(5)

        SUM   =                     TMP(I  ,J-1)    + TMP(I  ,J  ) +
     1          TMP(I+1,J-2)    + TMP(I+1,J-1)    + TMP(I+1,J  ) +
     2          TMP(I+2,J-2)    + TMP(I+2,J-1)

        SUMSQ =                     TMP(I  ,J-1)**2 + TMP(I  ,J  )**2 +
     1          TMP(I+1,J-2)**2 + TMP(I+1,J-1)**2 + TMP(I+1,J  )**2 +
     2          TMP(I+2,J-2)**2 + TMP(I+2,J-1)**2

        AVG( 5 ) = SUM / 7

        VAR( 5 ) = ( ( 7 * SUMSQ ) - ( SUM ** 2 ) ) / 42
```

```
*       Compute average and variance of pixels in sixth neighborhood
*       ( center left region )

  260   IF( I .EQ. 1 .OR. I .EQ. M .OR. J .LE. 2 ) THEN
        VAR( 6 )   65536.0
        GO TO 270
        END IF

*       Compute AVG(6) and VAR(6)

        SUM     TMP(I-1,J-2)    + TMP(I-1,J-1)    +
     1          TMP(I  ,J-2)    + TMP(I  ,J-1)    + TMP(I  ,J  )    +
     2          TMP(I+1,J-2)    + TMP(I+1,J-1)

        SUMSQ   TMP(I-1,J-2)**2 + TMP(I-1,J-1)**2 +
     1          TMP(I  ,J-2)**2 + TMP(I  ,J-1)**2 + TMP(I  ,J  )**2 +
     2          TMP(I+1,J-2)**2 + TMP(I+1,J-1)**2

        AVG( 6 )   SUM / 7

        VAR( 6 ) = ( ( 7 * SUMSQ )   ( SUM ** 2 ) ) / 42

*       Compute average and variance of pixels in seventh neighborhood
*       ( upper left region )

  270   IF( I .LE. 2 .OR. J .LE. 2 ) THEN
        VAR( 7 ) = 65536.0
        GO TO 280
        END IF

*       Compute AVG(7) and VAR(7)

        SUM     TMP(I-2,J-2)    + TMP(I-2,J-1)    +
     1          TMP(I-1,J-2)    + TMP(I-1,J-1)    + TMP(I-1,J  )    +
     2                            TMP(I  ,J-1)    + TMP(I  ,J  )

        SUMSQ   TMP(I-2,J-2)**2 + TMP(I-2,J-1)**2 +
     1          TMP(I-1,J-2)**2 + TMP(I-1,J-1)**2 + TMP(I-1,J  )**2 +
     2                            TMP(I  ,J-1)**2 + TMP(I  ,J  )**2

        AVG( 7 ) = SUM / 7

        VAR( 7 ) = ( ( 7 * SUMSQ )   ( SUM ** 2 ) ) / 42

*       Compute average and variance of pixels in eighth neighborhood
*       ( upper center region )

  280   IF( I .LE. 2 .OR. J .EQ. 1 .OR. J .EQ. N ) THEN
        VAR( 8 ) = 65536.0
        GO TO 290
        END IF

*       Compute AVG(8) and VAR(8)

        SUM     TMP(I-2,J-1)    + TMP(I-2,J  )    + TMP(I-2,J+1)    +
     1          TMP(I-1,J-1)    + TMP(I-1,J  )    + TMP(I-1,J+1)    +
     2                            TMP(I  ,J  )

        SUMSQ = TMP(I-2,J-1)**2 + TMP(I-2,J  )**2 + TMP(I-2,J+1)**2 +
     1          TMP(I-1,J-1)**2 + TMP(I-1,J  )**2 + TMP(I-1,J+1)**2 +
     2                            TMP(I  ,J  )**2

        AVG( 8 ) = SUM / 7

        VAR( 8 )   ( ( 7 * SUMSQ )   ( SUM ** 2 ) ) / 42

*       Compute average and variance of pixels in ninth neighborhood
*       ( central region )
```

```
  290   IF( I .EQ. 1 .OR. I .EQ. M .OR. J .EQ. 1 .OR. J .EQ. N ) THEN
        VAR( 9 ) = 65536.0
        GO TO 295
        END IF

*       Compute AVG(9) and VAR(9)

        SUM   = TMP(I-1,J-1)    + TMP(I-1,J  )    + TMP(I-1,J+1)    +
     1          TMP(I  ,J-1)    + TMP(I  ,J  )    + TMP(I  ,J+1)    +
     2          TMP(I+1,J-1)    + TMP(I+1,J  )    + TMP(I+1,J+1)

        SUMSQ   TMP(I-1,J-1)**2 + TMP(I-1,J  )**2 + TMP(I-1,J+1)**2 +
     1          TMP(I  ,J-1)**2 + TMP(I  ,J  )**2 + TMP(I  ,J+1)**2 +
     2          TMP(I+1,J-1)**2 + TMP(I+1,J  )**2 + TMP(I+1,J+1)**2

        AVG( 9 ) = SUM / 9

        VAR( 9 )   ( ( 9 * SUMSQ )   ( SUM ** 2 ) ) / 72

*       Determine which neighborhood has lowest variance

  295   VARLO   65536.0
        DO  K = 1, 9
                IF( VAR( K ) .LT. VARLO ) THEN
                VARLO    VAR( K )
                REGION   K
                END IF
        END DO

*       Assign average gray level of corresponding neighborhood to
*       pixel of interest.

        AVE   AVG( REGION )
        OUT( I, J )   ININT( AVE )


                END DO                  ! End of a single pass of original
        END DO                  ! Nagao and Matsuyama algorithm.

        WRITE ( 6, * ) '      Fass',L,' completed.'

        IF( L .EQ. 5 ) GO TO 299

*       Transfer output image to temporary working array

        DO  J = 1, N
                DO  I = 1, M
                TMP( I, J ) = OUT( I, J )
                END DO
        END DO

        END DO          ! Carry on with next iteration of algorithm

*       Write filtered image to disk.

  299   CALL  FIXOUT( OUT, N, M )         ! Write image to disk.

99999   STOP

        END
```

```
*******************************************************************************
*                                                                             *
*       PROGRAM  IDENTIFICATION  :  NAGLAP.FOR                                *
*                ____________________________                                 *
*                                                                             *
*       This program performs local noise filtering on 8 bit images, up      *
*       to 512 by 512 pixels.                                                 *
*                                                                             *
*       One algorithm is used in this program:                                *
*                 Modified Nagao and Matsuyama                                *
*                                                                             *
*       In the modified versions of both algorithms, the calculation of       *
*       the range is substituted to that of the variance in each of the       *
*       neighborhoods surrounding the pixel of interest.                      *
*                                                                             *
*                                                                             *
*       REQUIRED SUBROUTINES :  PIXIN.FOR                                     *
*                               PIXOUT.FOR                                    *
*                                                                             *
*******************************************************************************
*                                                                             *
*                                Marc R. Lapointe                             *
*                M.S. Thesis,  Imaging  and  Photographic  Science            *
*                      Rochester Institute of Technology                      *
*                                                                             *
*******************************************************************************
*                                                                             *
*       VARIABLE  IDENTIFICATION                                              *
*                 ______________________                                      *
*                                                                             *
*       IM      =  Array with digitized picture values                        *
*       TMP, AR =  Temporary working arrays                                   *
*       OUT     =  Output array with filtered image                           *
*       AVG(n)  =  Average image gray level in the nth neighborhood           *
*       VAR(n)     Variance in image gray levels in the nth neighborhood      *
*       RANGE(n)=  Range of gray levels in the nth neighborhood               *
*                                                                             *
*******************************************************************************
*                                                                             *
*       TYPE DECLARATION AND STORAGE ALLOCATION                               *
*                 ____________________________________                        *
*                                                                             *
      INTEGER*2   IM(512,512), OUT(512,512), AR(512,512)
      INTEGER*2   REGION, MINI, MAXI
      INTEGER*4   TMP(512,512), SUM, SUMSQ
      REAL*4      AVG( 9 ), VAR( 9 ), RANGE( 9 ), RANLO
      PARAMETER   ( N   512, M   512 )
*                                                                             *
*********                                                       ***********
*                                                                             *
*       COMPUTATION  BLOCK                                                    *
*                    _____________                                            *
*                                                                             *

*       Read in the image to be filtered
*       ________________________________

    1 WRITE ( 6, * ) ' '
      WRITE ( 6, * ) ' INPUT OF IMAGE TO BE FILTERED'

      CALL  PIXIN( IM, N, M )

*-------------------------------------------------------------------------------
*
*       Run modified Nagao and Matsuyama algorithm
*       __________________________________________

  300 WRITE ( 6, * ) ' '
```

```
      WRITE ( 6, * ) ' Now filtering the image with the '
      WRITE ( 6, * ) ' modified Nagao and Matsuyama algorithm.'
      WRITE ( 6, * ) '      Please be patient.  Running...'

*     Transfer original image to temporary working array to be
*     used by the algorithm.

      DO  J   1, N
              DO  I   1, M
              AR( I, J ) - IM( I, J )
              END DO
      END DO

*     Begin iterative process.

      DO  L   1, 5              ! Iterate filter 5 times.

*     Perform filtering on transferred image

      DO  J   1, N              ! Start scanning of image
              DO  I   1, M      ! pixel by pixel.

*     Compute range of pixels in first neighborhood
*     ( upper right corner )

  310 IF( I .LE. 2 .OR. J .GT. ( N   2 ) ) THEN
      RANGE( 1 )   256.0
      GO TO 320
      END IF

      MINI   AR(I-2,J+1)
      MAXI = AR(I-2,J+1)

      IF ( AR(I-2,J+2) .LT. MINI) THEN
        MINI   AR(I-2,J+2)
      ELSE IF ( AR(I-2,J+2) .GT. MAXI) THEN
        MAXI   AR(I-2,J+2)
      END IF

      IF ( AR(I-1,J) .LT. MINI) THEN
        MINI   AR(I-1,J)
      ELSE IF ( AR(I-1,J) .GT. MAXI) THEN
        MAXI   AR(I-1,J)
      END IF

      IF ( AR(I-1,J+1) .LT. MINI) THEN
        MINI   AR(I-1,J+1)
      ELSE IF ( AR(I-1,J+1) .GT. MAXI) THEN
        MAXI   AR(I-1,J+1)
      END IF

      IF ( AR(I-1,J+2) .LT. MINI) THEN
        MINI = AR(I-1,J+2)
      ELSE IF ( AR(I-1,J+2) .GT. MAXI) THEN
        MAXI = AR(I-1,J+2)
      END IF

      IF ( AR(I,J) .LT. MINI) THEN
        MINI = AR(I,J)
      ELSE IF ( AR(I,J) .GT. MAXI) THEN
        MAXI = AR(I,J)
      END IF

      IF ( AR(I,J+1) .LT. MINI) THEN
        MINI = AR(I,J+1)
      ELSE IF ( AR(I,J+1) .GT. MAXI) THEN
        MAXI = AR(I,J+1)
      END IF
```

```
          RANGE( 1 ) = ( MAXI   MINI ) / 2.704

*         Compute range of pixels in second neighborhood
*         ( center right region )

  320     IF( I .EQ. 1 .OR. I .EQ. M .OR. J .GT. ( N   2 ) ) THEN
          RANGE( 2 ) = 256
          GO TO 330
          END IF

          MINI   AR(I-1,J+1)
          MAXI   AR(I-1,J+1)

          IF ( AR(I-1,J+2) .LT. MINI) THEN
            MINI = AR(I-1,J+2)
          ELSE IF ( AR(I-1,J+2) .GT. MAXI) THEN
            MAXI   AR(I-1,J+2)
          END IF

          IF ( AR(I,J) .LT. MINI) THEN
            MINI = AR(I,J)
          ELSE IF ( AR(I,J) .GT. MAXI) THEN
            MAXI   AR(I,J)
          END IF

          IF ( AR(I,J+1) .LT. MINI) THEN
            MINI = AR(I,J+1)
          ELSE IF ( AR(I,J+1) .GT. MAXI) THEN
            MAXI = AR(I,J+1)
          END IF

          IF ( AR(I,J+2) .LT. MINI) THEN
            MINI   AR(I,J+2)
          ELSE IF ( AR(I,J+2) .GT. MAXI) THEN
            MAXI = AR(I,J+2)
          END IF

          IF ( AR(I+1,J+1) .LT. MINI) THEN
            MINI   AR(I+1,J+1)
          ELSE IF ( AR(I+1,J+1) .GT. MAXI) THEN
            MAXI   AR(I+1,J+1)
          END IF

          IF ( AR(I+1,J+2) .LT. MINI) THEN
            MINI = AR(I+1,J+2)
          ELSE IF ( AR(I+1,J+2) .GT. MAXI) THEN
            MAXI   AR(I+1,J+2)
          END IF

          RANGE( 2 )   ( MAXI   MINI ) / 2.704

*         Compute range of pixels in third neighborhood
*         ( lower right region )

  330     IF( I .GT. ( M   2 ) .OR. J .GT. ( N   2 ) ) THEN
          RANGE( 3 )   256
          GO TO 340
          END IF

          MINI = AR(I,J)
          MAXI = AR(I,J)

          IF ( AR(I,J+1) .LT. MINI) THEN
            MINI = AR(I,J+1)
          ELSE IF ( AR(I,J+1) .GT. MAXI) THEN
            MAXI = AR(I,J+1)
          END IF
```

```
          IF ( AR(I+1,J) .LT. MINI) THEN
            MINI = AR(I+1,J)
          ELSE IF ( AR(I+1,J) .GT. MAXI) THEN
            MAXI   AR(I+1,J)
          END IF

          IF ( AR(I+1,J+1) .LT. MINI) THEN
            MINI   AR(I+1,J+1)
          ELSE IF ( AR(I+1,J+1) .GT. MAXI) THEN
            MAXI   AR(I+1,J+1)
          END IF

          IF ( AR(I+1,J+2) .LT. MINI) THEN
            MINI   AR(I+1,J+2)
          ELSE IF ( AR(I+1,J+2) .GT. MAXI) THEN
            MAXI   AR(I+1,J+2)
          END IF

          IF ( AR(I+2,J+1) .LT. MINI) THEN
            MINI   AR(I+2,J+1)
          ELSE IF ( AR(I+2,J+1) .GT. MAXI) THEN
            MAXI   AR(I+2,J+1)
          END IF

          IF ( AR(I+2,J+2) .LT. MINI) THEN
            MINI = AR(I+2,J+2)
          ELSE IF ( AR(I+2,J+2) .GT. MAXI) THEN
            MAXI   AR(I+2,J+2)
          END IF

          RANGE( 3 )   ( MAXI   MINI ) / 2.704

*         Compute range of pixels in forth neighborhood
*         ( lower center region )

    340   IF( I .GT. ( M   2 ) .OR. J .EQ. 1 .OR. J .EQ. N ) THEN
          RANGE( 4 ) = 256
          GO TO 350
          END IF

          MINI   AR(I,J)
          MAXI   AR(I,J)

          IF ( AR(I+1,J-1) .LT. MINI) THEN
            MINI = AR(I+1,J-1)
          ELSE IF ( AR(I+1,J-1) .GT. MAXI) THEN
            MAXI = AR(I+1,J-1)
          END IF

          IF ( AR(I+1,J) .LT. MINI) THEN
            MINI   AR(I+1,J)
          ELSE IF ( AR(I+1,J) .GT. MAXI) THEN
            MAXI   AR(I+1,J)
          END IF

          IF ( AR(I+1,J+1) .LT. MINI) THEN
            MINI = AR(I+1,J+1)
          ELSE IF ( AR(I+1,J+1) .GT. MAXI) THEN
            MAXI   AR(I+1,J+1)
          END IF

          IF ( AR(I+2,J-1) .LT. MINI) THEN
            MINI   AR(I+2,J-1)
          ELSE IF ( AR(I+2,J-1) .GT. MAXI) THEN
            MAXI   AR(I+2,J-1)
          END IF

          IF ( AR(I+2,J) .LT. MINI) THEN
```

```
          MINI = AR(I+2,J)
        ELSE IF ( AR(I+2,J) .GT. MAXI) THEN
          MAXI = AR(I+2,J)
        END IF

        IF ( AR(I+2,J+1) .LT. MINI) THEN
          MINI   AR(I+2,J+1)
        ELSE IF ( AR(I+2,J+1) .GT. MAXI) THEN
          MAXI = AR(I+2,J+1)
        END IF

        RANGE( 4 )   ( MAXI   MINI ) / 2.704

*       Compute range of pixels in fifth neighborhood
*       ( lower left region )

  350   IF( I .GT. ( M   2 ) .OR. J .LE. 2 ) THEN
        RANGE( 5 ) = 256
        GO TO 360
        END IF

        MINI   AR(I,J-1)
        MAXI   AR(I,J-1)

        IF ( AR(I,J) .LT. MINI) THEN
          MINI = AR(I,J)
        ELSE IF ( AR(I,J) .GT. MAXI) THEN
          MAXI = AR(I,J)
        END IF

        IF ( AR(I+1,J-2) .LT. MINI) THEN
          MINI   AR(I+1,J-2)
        ELSE IF ( AR(I+1,J-2) .GT. MAXI) THEN
          MAXI   AR(I+1,J-2)
        END IF

        IF ( AR(I+1,J-1) .LT. MINI) THEN
          MINI   AR(I+1,J-1)
        ELSE IF ( AR(I+1,J-1) .GT. MAXI) THEN
          MAXI   AR(I+1,J-1)
        END IF

        IF ( AR(I+1,J) .LT. MINI) THEN
          MINI   AR(I+1,J)
        ELSE IF ( AR(I+1,J) .GT. MAXI) THEN
          MAXI = AR(I+1,J)
        END IF

        IF ( AR(I+2,J-2) .LT. MINI) THEN
          MINI   AR(I+2,J-2)
        ELSE IF ( AR(I+2,J-2) .GT. MAXI) THEN
          MAXI   AR(I+2,J-2)
        END IF

        IF ( AR(I+2,J-1) .LT. MINI) THEN
          MINI   AR(I+2,J-1)
        ELSE IF ( AR(I+2,J-1) .GT. MAXI) THEN
          MAXI = AR(I+2,J-1)
        END IF

        RANGE( 5 ) = ( MAXI   MINI ) / 2.704

*       Compute range of pixels in sixth neighborhood
*       ( center left region )

  360   IF( I .EQ. 1 .OR. I .EQ. M .OR. J .LE. 2 ) THEN
        RANGE( 6 ) = 256
        GO TO 370
```

```
      END IF

      MINI = AR(I-1,J-2)
      MAXI = AR(I-1,J-2)

      IF ( AR(I-1,J-1) .LT. MINI) THEN
        MINI   AR(I-1,J-1)
      ELSE IF ( AR(I-1,J-1) .GT. MAXI) THEN
        MAXI   AR(I-1,J-1)
      END IF

      IF ( AR(I,J-2) .LT. MINI) THEN
        MINI = AR(I,J-2)
      ELSE IF ( AR(I,J-2) .GT. MAXI) THEN
        MAXI   AR(I,J-2)
      END IF

      IF ( AR(I,J-1) .LT. MINI) THEN
        MINI = AR(I,J-1)
      ELSE IF ( AR(I,J-1) .GT. MAXI) THEN
        MAXI = AR(I,J-1)
      END IF

      IF ( AR(I,J) .LT. MINI) THEN
        MINI   AR(I,J)
      ELSE IF ( AR(I,J) .GT. MAXI) THEN
        MAXI = AR(I,J)
      END IF

      IF ( AR(I+1,J-2) .LT. MINI) THEN
        MINI   AR(I+1,J-2)
      ELSE IF ( AR(I+1,J-2) .GT. MAXI) THEN
        MAXI   AR(I+1,J-2)
      END IF

      IF ( AR(I+1,J-1) .LT. MINI) THEN
        MINI = AR(I+1,J-1)
      ELSE IF ( AR(I+1,J-1) .GT. MAXI) THEN
        MAXI   AR(I+1,J-1)
      END IF

      RANGE( 6 )   ( MAXI   MINI ) / 2.704

*     Compute range of pixels in seventh neighborhood
*     ( upper left region )

  370 IF( I .LE. 2 .OR. J .LE. 2 ) THEN
      RANGE( 7 ) - 256
      GO TO 380
      END IF

      MINI = AR(I-2,J-2)
      MAXI   AR(I-2,J-2)

      IF ( AR(I-2,J-1) .LT. MINI) THEN
        MINI = AR(I-2,J-1)
      ELSE IF ( AR(I-2,J-1) .GT. MAXI) THEN
        MAXI = AR(I-2,J-1)
      END IF

      IF ( AR(I-1,J-2) .LT. MINI) THEN
        MINI   AR(I-1,J-2)
      ELSE IF ( AR(I-1,J-2) .GT. MAXI) THEN
        MAXI   AR(I-1,J-2)
      END IF

      IF ( AR(I-1,J-1) .LT. MINI) THEN
        MINI = AR(I-1,J-1)
```

```
      ELSE IF ( AR(I-1,J-1) .GT. MAXI) THEN
        MAXI   AR(I-1,J-1)
      END IF

      IF ( AR(I-1,J) .LT. MINI) THEN
        MINI   AR(I-1,J)
      ELSE IF ( AR(I-1,J) .GT. MAXI) THEN
        MAXI = AR(I-1,J)
      END IF

      IF ( AR(I,J-1) .LT. MINI) THEN
        MINI = AR(I,J-1)
      ELSE IF ( AR(I,J-1) .GT. MAXI) THEN
        MAXI   AR(I,J-1)
      END IF

      IF ( AR(I,J) .LT. MINI) THEN
        MINI   AR(I,J)
      ELSE IF ( AR(I,J) .GT. MAXI) THEN
        MAXI   AR(I,J)
      END IF

      RANGE( 7 ) = ( MAXI   MINI ) / 2.704

*     Compute range of pixels in eighth neighborhood
*     ( upper center region )

  380 IF( I .LE. 2 .OR. J .EQ. 1 .OR. J .EQ. N ) THEN
      RANGE( 8 ) = 256
      GO TO 390
      END IF

      MINI = AR(I-2,J-1)
      MAXI   AR(I-2,J-1)

      IF ( AR(I-2,J) .LT. MINI) THEN
        MINI   AR(I-2,J)
      ELSE IF ( AR(I-2,J) .GT. MAXI) THEN
        MAXI   AR(I-2,J)
      END IF

      IF ( AR(I-2,J+1) .LT. MINI) THEN
        MINI   AR(I-2,J+1)
      ELSE IF ( AR(I-2,J+1) .GT. MAXI) THEN
        MAXI   AR(I-2,J+1)
      END IF

      IF ( AR(I-1,J-1) .LT. MINI) THEN
        MINI = AR(I-1,J-1)
      ELSE IF ( AR(I-1,J-1) .GT. MAXI) THEN
        MAXI   AR(I-1,J-1)
      END IF

      IF ( AR(I-1,J) .LT. MINI) THEN
        MINI = AR(I-1,J)
      ELSE IF ( AR(I-1,J) .GT. MAXI) THEN
        MAXI = AR(I-1,J)
      END IF

      IF ( AR(I-1,J+1) .LT. MINI) THEN
        MINI = AR(I-1,J+1)
      ELSE IF ( AR(I-1,J+1) .GT. MAXI) THEN
        MAXI   AR(I-1,J+1)
      END IF

      IF ( AR(I,J) .LT. MINI) THEN
        MINI   AR(I,J)
      ELSE IF ( AR(I,J) .GT. MAXI) THEN
```

```
          MAXI   AR(I,J)
        END IF

        RANGE( 8 )   ( MAXI   MINI ) / 2.704

*       Compute range of pixels in ninth neighborhood
*       ( central region )

  390   IF( I .EQ. 1 .OR. I .EQ. M .OR. J .EQ. 1 .OR. J .EQ. N ) THEN
        RANGE( 9 )   256
        GO TO 400
        END IF

        MINI   AR(I-1,J-1)
        MAXI   AR(I-1,J-1)

        IF ( AR(I-1,J) .LT. MINI) THEN
          MINI = AR(I-1,J)
        ELSE IF ( AR(I-1,J) .GT. MAXI) THEN
          MAXI   AR(I-1,J)
        END IF

        IF ( AR(I-1,J+1) .LT. MINI) THEN
          MINI = AR(I-1,J+1)
        ELSE IF ( AR(I-1,J+1) .GT. MAXI) THEN
          MAXI = AR(I-1,J+1)
        END IF

        IF ( AR(I,J-1) .LT. MINI) THEN
          MINI = AR(I,J-1)
        ELSE IF ( AR(I,J-1) .GT. MAXI) THEN
          MAXI   AR(I,J-1)
        END IF

        IF ( AR(I,J) .LT. MINI) THEN
          MINI   AR(I,J)
        ELSE IF ( AR(I,J) .GT. MAXI) THEN
          MAXI   AR(I,J)
        END IF

        IF ( AR(I,J+1) .LT. MINI) THEN
          MINI   AR(I,J+1)
        ELSE IF ( AR(I,J+1) .GT. MAXI) THEN
          MAXI   AR(I,J+1)
        END IF

        IF ( AR(I+1,J-1) .LT. MINI) THEN
          MINI = AR(I+1,J-1)
        ELSE IF ( AR(I+1,J-1) .GT. MAXI) THEN
          MAXI = AR(I+1,J-1)
        END IF

        IF ( AR(I+1,J) .LT. MINI) THEN
          MINI = AR(I+1,J)
        ELSE IF ( AR(I+1,J) .GT. MAXI) THEN
          MAXI = AR(I+1,J)
        END IF

        IF ( AR(I+1,J+1) .LT. MINI) THEN
          MINI = AR(I+1,J+1)
        ELSE IF ( AR(I+1,J+1) .GT. MAXI) THEN
          MAXI = AR(I+1,J+1)
        END IF

        RANGE( 9 )   ( MAXI   MINI ) / 2.970

*       Determine which neighborhood has lowest range
```

```
400   RANLO   RANGE( 1 )
      REGION= 1
      DO  K = 2, 9
              IF( RANGE( K ) .LT. RANLO ) THEN
              RANLO = RANGE( K )
              REGION   K
              END IF
      END DO

      IF( REGION .EQ. 1 ) THEN
      OUT(I,J)=ININT((             AR(I-2,J+1)+AR(I-2,J+2)+
     1                    AR(I-1,J)+AR(I-1,J+1)+AR(I-1,J+2)+
     2                    AR(I  ,J)+AR(I  ,J+2)              )/7.0)

      ELSE IF( REGION .EQ. 2 ) THEN
      OUT(I,J)=ININT((               AR(I-1,J+1)+AR(I-1,J+2)+
     1                    AR(I  ,J  )+AR(I  ,J+1)+AR(I  ,J+2)+
     2                      AR(I+1,J+1)+AR(I+1,J+2))/7.0)

      ELSE IF( REGION .EQ. 3 ) THEN
      OUT(I,J)=ININT((   AR(I  ,J  )+AR(I  ,J+1)+
     1                    AR(I+1,J  )+AR(I+1,J+1)+AR(I+1,J+2)+
     2                      AR(I+2,J+1)+AR(I+2,J+2))/7.0)

      ELSE IF( REGION .EQ. 4 ) THEN
      OUT(I,J)=ININT((               AR(I  ,J  )+
     1                    AR(I+1,J-1)+AR(I+1,J  )+AR(I+1,J+1)+
     2                    AR(I+2,J-1)+AR(I+2,J  )+AR(I+2,J+1))/7.0)

      ELSE IF( REGION .EQ. 5 ) THEN
      OUT(I,J)=ININT((               AR(I  ,J-1)+AR(I  ,J  )+
     1                    AR(I+1,J-2)+AR(I+1,J-1)+AR(I+1,J  )+
     2                    AR(I+2,J-2)+AR(I+2,J-1)              )/7.0)

      ELSE IF( REGION .EQ. 6 ) THEN
      OUT(I,J)=ININT((   AR(I-1,J-2)+AR(I-1,J-1)+
     1                    AR(I  ,J-2)+AR(I  ,J-1)+AR(I  ,J  )+
     2                  AR(I+1,J-2)+AR(I+1,J-1)             )/7.0)

      ELSE IF( REGION .EQ. 7 ) THEN
      OUT(I,J)=ININT((   AR(I-2,J-2)+AR(I-2,J-1)+
     1                   AR(I-1,J-2)+AR(I-1,J-1)+AR(I-1,J  )+
     2                              AR(I  ,J-1)+AR(I  ,J  ))/7.0)

      ELSE IF( REGION .EQ. 8 ) THEN
      OUT(I,J)=ININT((   AR(I-2,J-1)+AR(I-2,J  )+AR(I-2,J+1)+
     1                   AR(I-1,J-1)+AR(I-1,J  )+AR(I-1,J+1)+
     2                               AR(I  ,J  )             )/7.0)

      ELSE IF( REGION .EQ. 9 ) THEN
      OUT(I,J)=ININT((   AR(I-1,J-1)+AR(I-1,J  )+AR(I-1,J+1)+
     1                   AR(I  ,J-1)+AR(I  ,J  )+AR(I  ,J+1)+
     2                   AR(I+1,J-1)+AR(I+1,J  )+AR(I+1,J+1))/9.0)
      END IF


410           END DO                    ! End of a single pass of modified
      END DO                      ! Nagao and Matsuyama algorithm.

      WRITE ( 6, * ) '      Pass ', L, ' completed.'

      IF( L .EQ. 5 ) GO TO 420

*     Transfer output image to temporary working array

      DO  J   1, N
              DO  I = 1, M
              AR( I, J ) = OUT( I, J )
```

```
            END DO
        END DO

        END DO          ! Carry on with next iteration of algorithm
*       Write filtered image to disk.
  420   CALL  PIXOUT( OUT, N, M )          ! Write image to disk.
99999   STOP
        END
```

APPENDIX 7

SAMPLE OF FILTERED IMAGES

(a)

(b)

(c)

(d)

The noisy image shown in Appendix 5 filtered with:
(a) the Tomita/Tsuji algorithm,
(b) the modified Tomita/Tsuji algorithm,
(c) the Nagao/Matsuyama algorithm, and
(d) the modified Nagao/Matsuyama algorithm.

APPENDIX 8

PRELIMINARY RESULTS: EVALUATION SHEETS

SUBJECTIVE EVALUATION
for M.S. THESIS
MARC R. LAPOINTE

RESULTS

OBSERVER'S NAME: John Francis DATE: 1/6/86

IMAGE SERIES : 9

	Noise smoothing ability		Preservation of subtle details		Immunity from shape distortion		Retention of step edges	
	Choice	Rating	Choice	Rating	Choice	Rating	Choice	Rating
A vs B	B	6	A	8	A	2	B	6
A vs C	C	4	A	5	A	3	C	4
A vs D	D	4	A	5	A	5	D	4
A vs E	F	9	A	10	A	9	A	7
B vs C	B	1	C	3	C	4	B	4
B vs D	D	2	D	2	D	4	B	5
B vs E	B	1	B	1	B	5	B	6
C vs D	C	2	C	3	C	1	D	2
C vs E	E	1	C	6	C	5	C	2
D vs E	E	3	D	2	D	7	D	4

SUBJECTIVE EVALUATION
M.S. THESIS
MARC R. LAPOINTE

INSTRUCTIONS

For each pair of images presented, select the image you perceive to be the best according to the four criteria listed below.

Also rate how much apart the two images are (still according to the same criteria) on a scale of zero to ten. An increase in the rating signifies a larger difference between the images.

You must make a selection. If both images appear the same, select one and give it a low rating (zero accepted).

EVALUATION CRITERIA

1. EFFECTIVENESS AT NOISE SMOOTHING - the reduction in noise variance in a flat region of the image.

2. PRESERVATION OF SUBTLE DETAILS - ability to retain highly distinguishable subtle details.

3. IMMUNITY FROM SHAPE DISTORTION - the algorithm may create significant distortions as well as artifacts (presence of a structure in the enhanced image with no ground truth or basis for existence other than being artificially induced by a computer algorithmic process.

4. RETENTION OF STEP EDGES AND SHARPENING OF RAMP EDGES - ability to sharpen blurry edges.

APPENDIX 9

COMPUTER PROGRAM: DATA REDUCTION

```
************************************************************************
*                                                                      *
*       PROGRAM IDENTIFICATION: DATARED                                *
*       ___________________________                                    *
*                                                                      *
*       THIS PROGRAM PERFORMS DATA REDUCTION ON RAW STATISTICAL        *
*       DISTANCES OBTAINED FROM A SUBJECTIVE ANALYSIS ON IMAGES        *
*       PRODUCED DURING M.S. DEGREE THESIS BY MARC. R. LAPOINTE        *
*
*       TYPE DECLARATION AND STORAGE ALLOCATION
*       ______________________________________
*
        REAL            RAWDIST(5,15), CORREL(15,15), CORRSTATS(2,15)
        REAL            REGRESS(3,15), ADJDIST(5,15), MEANS (5,4)
        REAL            SX, SY, SXY, SX2, SY2, SLOPE, INTCP, AVG, SDEV
        REAL            SHIFT, OBS(15), T, TTEST(15)
        INTEGER         N1, N2, N3, REF, NSY,TOM,TLAP,NAG,NLAP
        CHARACTER*20    FILENAME
        CHARACTER*1     ANS
        DATA            OBS/15*1.0/
        DATA            TTEST/8.000,4.303,2.776,2.447,2.306,2.228,2.179,
    1                   2.145,2.120,2.101,2.086,2.074,2.064,2.056,2.048/
*
*****************
*
*       Read in name of file with raw distances data

    5   WRITE ( 6,100 )
  100   FORMAT( /,1X,10(1H>),'  What is the name of the file with the',
    1                        ' raw distances data ? : ', $ )
        READ  ( 5,'(A)', END=5 ) FILENAME

*       Open file at logical unit 1
*       ___________________________

        OPEN  ( 1, FILE=FILENAME, ACCESS='SEQUENTIAL', STATUS='OLD' )

*       Read in raw distances into matrix
*       _________________________________

        DO  J=1,15
                DO  I=1,5
                  READ(1,*) RAWDIST(I,J)
                END DO
        END DO

*       Read in order of each image used in paired comparison
*       _____________________________________________________

        READ  (1, * ) NSY, TOM, TLAP, NAG, NLAP

*       Close open file at logical unit 1
*       _________________________________

        CLOSE ( 1, STATUS='KEEP' )

*       Write raw distance matrix
*       _________________________
```

```
      WRITE (6,101)
  101 FORMAT(//, 1X, 50(1H*), '  OBSERVERS'' RAW  DISTANCES  ',
     1            50(1H*) )
      WRITE (6,105)
  105 FORMAT(/,1X,'OBSERVERS:',4X,'1',7X,'2',7X,'3',7X,'4',7X,'5',7X,
     1  '6',7X,'7',7X,'8',7X,'9',6X,'10',6X,'11',6X,'12',6X,'13',6X,
     2  '14',6X,'15', / )
      WRITE(6,'(1X,A,1X,15(3X,F5.1))')'Image A',(RAWDIST(1,J),J=1,15)
      WRITE(6,'(1X,A,1X,15(3X,F5.1))')'Image B',(RAWDIST(2,J),J=1,15)
      WRITE(6,'(1X,A,1X,15(3X,F5.1))')'Image C',(RAWDIST(3,J),J=1,15)
      WRITE(6,'(1X,A,1X,15(3X,F5.1))')'Image D',(RAWDIST(4,J),J=1,15)
      WRITE(6,'(1X,A,1X,15(3X,F5.1))')'Image E',(RAWDIST(5,J),J=1,15)


*************************************************************************
*                                                                       *
*       STAGE 1 : DETERMINE CORRELATION COEFFICIENTS FOR EACH SET       *
*       OF RAW DISTANCES WITH REGARD TO EVERY OTHER SET IN DESCEN-      *
*       DING ORDER                                                      *
*                                                                       *
*************************************************************************

      DO  N1= 1,14
        CORREL(N1,N1) = 1.0
              DO  N2 = (N1+1),15
                SX    0.0
                SY    0.0
                SXY   0.0
                SX2   0.0
                SY2   0.0
                    DO  N3   1,5
                      SX    SX  + RAWDIST(N3,N1)
                      SY    SY  + RAWDIST(N3,N2)
                      SXY   SXY + RAWDIST(N3,N1) * RAWDIST(N3,N2)
                      SX2   SX2 + RAWDIST(N3,N1) ** 2
                      SY2   SY2 + RAWDIST(N3,N2) ** 2
                    END DO
                CORREL(N1,N2)   ( 5*SXY   SX*SY ) /
    1                        SQRT(( 5*SX2-SX**2) * ( 5*SY2-SY**2 ))
                CORREL(N2,N1) = CORREL(N1,N2)
              END DO
      END DO
      CORREL(15,15)   1.0

*     Write correlation matrix
*     ------------------------

      WRITE (6,110)
  110 FORMAT(//, 1X, 41(1H*), '  LINEAR CORRELATION FACTORS BETWEEN '
     1 'OBSERVERS  ', 40(1H*) )
      WRITE (6,105)
      DO  N1=1,15
        WRITE (6,'(4X,I2,4X,15(1X,F7.4))') N1, (CORREL(N1,J),J=1,15)
      END DO
*************************************************************************
*                                                                       *
*       STAGE 2 : SELECT THE SET OF RAW DISTANCES WHICH IS THE          *
*       MOST CORRELATED WITH AS MANY OTHER SETS AS POSSIBLE, AS         *
*       THE REFERENCE DATA SET                                          *
*                                                                       *
*************************************************************************


      DO  N1   1,15
        SX  = 0.0
        SX2 = 0.0
              DO  N2   1,15
                IF( N2.EQ.N1) GO TO 20
```

```
                SX    SX  + CORREL(N1,N2)
                SX2   SX2 + CORREL(N1,N2)**2
   20         END DO
        CORRSTATS(1,N1) = SX/14
        CORRSTATS(2,N1) = SQRT( (14*SX2   SX**2) / (14*13) )
      END DO

      WRITE (6,200)
  200 FORMAT(//,1X, 52(1H*),' CORRELATION STATISTICSS ',52(1H*),/,
     1 35X,'( Average correlation with other observers and standard',
     2 ' deviation)' )
      WRITE(6,105)
      WRITE(6,'(3X,A,4X,15(1X,F7.4))')'AVG', (CORRSTATS(1,J),J=1,15)
      WRITE(6,'(3X,A,3X,15(1X,F7.4))')'SDEV',(CORRSTATS(2,J),J=1,15)

*     Chose one data set as the reference set
*     ---------------------------------------

      WRITE (6,210)
  210 FORMAT(///,1X,10(1H>),'  Which data set do you wish to chose',
     1          ' as the reference data set',/,13X,'for the remainder',
     2          ' of the calculations ? (enter observer number): ', $)
      READ  ( 5, *) REF

*************************************************************************
*                                                                       *
*     STAGE 3 : ADJUST ALL RAW DATA SET TO THE CHOSEN REFERENCE         *
*     DATA SET                                                          *
*                                                                       *
*************************************************************************

      SX  = 0.0
      SX2   0.0

      DO  N3   1,5
        SX    SX  + RAWDIST(N3,REF)
        SX2 = SX2 + RAWDIST(N3,REF)**2
      END DO

      DO  N2   1,15
        REGRESS(1,N2) = CORREL(REF,N2)
        SY    0.0
        SXY   0.0
        SY2   0.0
              DO  N1 = 1,5
                SY    SY  + RAWDIST(N1,N2)
                SXY   SXY + RAWDIST(N1,N2) * RAWDIST(N1,REF)
                SY2   SY2 + RAWDIST(N1,N2) ** 2
              END DO
        SLOPE   ( 5*SXY   SX*SY )/( 5*SX2   SX**2 )
        INTCP   SY/5   ( SLOPE*SX / 5 )
        REGRESS(2,N2) = SLOPE
        REGRESS(3,N2) = INTCP

*             Adjust distances to reference observer
*             --------------------------------------

              DO  N1   1,5
                ADJDIST(N1,N2)   ( RAWDIST(N1,N2)   INTCP ) / SLOPE
              END DO
      END DO

*     Write linear regression and adjusted distances matrices
*     -------------------------------------------------------

      WRITE (6,300)
  300 FORMAT(///,1X,50(1H*),' LINEAR REGRESSION PARAMETERS ',49(1H*),
     1    /,40X,'( Correlation with ref., slope, and intercept )' )
```

```
      WRITE (6,105)
      WRITE (6,'(2X,A5,3X,15(1X,F7.4))')'CORR ',(REGRESS(1,J),J=1,15)
      WRITE (6,'(2X,A5,3X,15(1X,F7.4))')'SLOPE',(REGRESS(2,J),J=1,15)
      WRITE (6,'(2X,A5,3X,15(1X,F7.4))')'INTCP',(REGRESS(3,J),J=1,15)
      WRITE (6,310)
  310 FORMAT(////////,1X,54(1H*),'  ADJUSTED DISTANCES  ',53(1H*))
      WRITE(6,105)
      WRITE(6,'(1X,A,1X,15(1X,F7.2))')'Image A',(ADJDIST(1,J),J=1,15)
      WRITE(6,'(1X,A,1X,15(1X,F7.2))')'Image B',(ADJDIST(2,J),J=1,15)
      WRITE(6,'(1X,A,1X,15(1X,F7.2))')'Image C',(ADJDIST(3,J),J=1,15)
      WRITE(6,'(1X,A,1X,15(1X,F7.2))')'Image D',(ADJDIST(4,J),J=1,15)
      WRITE(6,'(1X,A,1X,15(1X,F7.2))')'Image E',(ADJDIST(5,J),J=1,15)

C*********************************************************************
C*                                                                   *
C*      STAGE 4 : ELIMINATE DATA SETS THAT ARE OUTSIDE OF THREE      *
C*      STANDARD DEVIATIONS OF THE MEAN DISTANCE BEING EVALUATED     *
C*      (i.e. 99.9% OF THE DATA FOR A NORMAL DISTRIBUTION )          *
C*                                                                   *
C*********************************************************************

C       Compute average and std. dev. of adjusted distances
C       ---------------------------------------------------

   40 WRITE (6,400)
  400 FORMAT(//,1X,51(1H*),'  MEAN ADJUSTED DISTANCES  ',51(1H*),//,
     1 32X, 'Mean', 12X, 'Std Dev', 11X, 'LoLimit', 11X, 'HiLimit',/)
      DO  N1= 1,5
        N3  = 0
        SX    0.0
        SX2   0.0
              DO  N2=1,15
                N3  = N3  + OBS(N2)
                SX    SX  + ADJDIST(N1,N2)    * OBS(N2)
                SX2   SX2 + ADJDIST(N1,N2)**2 * OBS(N2)
              END DO
        MEANS(N1,1) = SX / N3
        MEANS(N1,2) = SQRT( ( N3*SX2   SX**2 ) / ( N3*(N3-1) ) )
        MEANS(N1,3) = MEANS(N1,1)   3*MEANS(N1,2)
        MEANS(N1,4) = MEANS(N1,1) + 3*MEANS(N1,2)
      END DO

      WRITE(6,'(8X,A,4X,4(8X,F10.4))')'Image A',(MEANS(1,J), J=1,4)
      WRITE(6,'(8X,A,4X,4(8X,F10.4))')'Image B',(MEANS(2,J), J=1,4)
      WRITE(6,'(8X,A,4X,4(8X,F10.4))')'Image C',(MEANS(3,J), J=1,4)
      WRITE(6,'(8X,A,4X,4(8X,F10.4))')'Image D',(MEANS(4,J), J=1,4)
      WRITE(6,'(8X,A,4X,4(8X,F10.4))')'Image E',(MEANS(5,J), J=1,4)

C       Eliminate unwanted data sets
C       ----------------------------

   41 WRITE (6,410)
  410 FORMAT(//,1X,10(1H>),'  Do you wish to eliminate data sets '
     1 '(due to negative correlation or outside 3 standard deviations '
     2 'limit) ? (Y/N) : ', $ )
      READ  (5,'(A)')  ANS
      IF(ANS.EQ.'Y'.OR.ANS.EQ.'y') GO TO 42
      IF(ANS.EQ.'N'.OR.ANS.EQ.'n') GO TO 45
      GO TO 41
   42 WRITE (6,420)
  420 FORMAT(/,1X,10(1H>),'  Which observer''s data set do you wish'
     1 ' to eliminate ? (enter integer 1-15) : ', $ )
      READ  (5,*)  N1
      OBS(N1) = 0.0
   43 WRITE (6,430)
  430 FORMAT(/,1X,10(1H>),'  Eliminate another one ? (Y/N) : ', $ )
      READ  (5,'(A)')  ANS
      IF(ANS.EQ.'Y'.OR.ANS.EQ.'y') GO TO 42
```

```
        IF(ANS.EQ.'N'.OR.ANS.EQ.'n') GO TO 40
        GO TO 43

*       Reorder and shift distances
*       ---------------------------

   45   WRITE (6,450)
  450   FORMAT(//,1X,48(1H*),' SHIFTED MEAN ADJUSTED DISTANCES ',48(1H*),
     1  /,54X,'Shifted to Noisy   0.0 ',//,76X,'Mean',11X,'Std Dev',/)

*       Scale mean distances to NSY=0.0

        SHIFT    MEANS(NSY,1)
        DO  N1   1,5
          MEANS(N1,1)   MEANS(N1,1)   SHIFT
        END DO

*       Write mean distances in reordered fashion

        WRITE(6,'(24X,A35,5X,2(10X,F7.4))')'NSY     (Noisy)
     1     ',(MEANS(NSY,J),J=1,2)
        WRITE(6,'(24X,A35,5X,2(10X,F7.4))')'T/T     (Tomita/Tsuji)
     1     ',(MEANS(TOM,J),J=1,2)
        WRITE(6,'(24X,A35,5X,2(10X,F7.4))')'Mod.T/T (Modified Tomita/Tsuj
     1i)   ',(MEANS(TLAP,J),J=1,2)
        WRITE(6,'(24X,A35,5X,2(10X,F7.4))')'N/M     (Nagao/Matsuyama)
     1     ',(MEANS(NAG,J),J=1,2)
        WRITE(6,'(24X,A35,5X,2(10X,F7.4))')'Mod.N/M (Modified Nagao/Matsu
     1yama)',(MEANS(NLAP,J),J=1,2)


*************************************************************************
*                                                                       *
*       STAGE 5 : PERFORM HYPOTESIS TESTING ON REMAINING DATA SETS      *
*                                                                       *
*************************************************************************

        WRITE (6,500)
  500   FORMAT(//,1X,55(1H*),' HYPOTHESIS TESTING ',54(1H*),/,30X,
     1  'Null Ho: u1 = u2 ;  Alternate H1: u1 # u2 ;   Significance ',
     2  'level Alpha = 0.05')

        WRITE (6,501) N3, TTEST(N3)
  501   FORMAT(/,40X,'t(0.025) value for ',I2,' observers is:    ',F5.2)

*       Test between noisy image and Tomita/Tsuji algorithm
*       ---------------------------------------------------

        T = ( MEANS(TOM,1)   MEANS(NSY,1) ) /
     1          SQRT( ( MEANS(TOM,2)**2 + MEANS(NSY,2)**2 ) / N3 )

*       Compare t value above with t(0.025) for N3 observers:

        IF( ABS(T).GT.TTEST(N3)) THEN
          WRITE (6,510) T
  510     FORMAT(/,25X,'Test between NSY     and T/T     images:      ',
     1    't  = ', F8.2, ' ;            Hypothesis REJECTED')
        ELSE
          WRITE (6,511) T
  511     FORMAT(/,25X,'Test between NSY     and T/T     images:      ',
     1    't    ', F8.2, ' ;              CANNOT REJECT')
        END IF

*       Test between noisy image and Nagao/Matsuyama algorithm
*       ------------------------------------------------------

        T   ( MEANS(NAG,1)   MEANS(NSY,1) ) /
     1          SQRT( ( MEANS(NAG,2)**2 + MEANS(NSY,2)**2 ) / N3 )
```

```
*       Compare t value above with t(0.025) for N3 observers:

        IF( ABS(T).GT.TTEST(N3)) THEN
          WRITE (6,520) T
  520     FORMAT(25X,'Test between NSY     and N/M     images:      ',
     1    't  = ', F8.2, ' ;            Hypothesis REJECTED')
        ELSE
          WRITE (6,521) T
  521     FORMAT(25X,'Test between NSY     and N/M     images:      ',
     1    't    '. F8.2, ' ;              CANNOT REJECT')
        END IF

*       Test between Tomita/Tsuji and Nagao/Matsuyama algorithms
*       ------------------------------------------------------------

        T = ( MEANS(NAG,1)   MEANS(TOM,1) ) /
     1          SQRT( ( MEANS(NAG,2)**2 + MEANS(TOM,2)**2 ) / N3 )

*       Compare t value above with t(0.025) for N3 observers:

        IF( ABS(T).GT.TTEST(N3)) THEN
          WRITE (6,530) T
  530     FORMAT(25X,'Test between T/T     and N/M     images:      ',
     1    't    ', F8.2, ' ;            Hypothesis REJECTED')
        ELSE
          WRITE (6,531) T
  531     FORMAT(25X,'Test between T/T     and N/M     images:      ',
     1    't    ', F8.2, ' ;              CANNOT REJECT')
        END IF

*       Test between noisy image and Modified Tomita/Tsuji algorithm
*       ------------------------------------------------------------

        T = ( MEANS(TLAP,1)   MEANS(NSY,1) ) /
     1          SQRT( ( MEANS(TLAP,2)**2 + MEANS(NSY,2)**2 ) / N3 )

*       Compare t value above with t(0.025) for N3 observers:

        IF( ABS(T).GT.TTEST(N3)) THEN
          WRITE (6,540) T
  540     FORMAT(25X,'Test between NSY     and Mod.T/T images:      ',
     1    't    ', F8.2, ' ;            Hypothesis REJECTED')
        ELSE
          WRITE (6,541) T
  541     FORMAT(25X,'Test between NSY     and Mod.T/T images:      ',
     1    't    ', F8.2, ' ;              CANNOT REJECT')
        END IF

*       Test between noisy image and Modified Nagao/Matsuyama alg.
*       ----------------------------------------------------------

        T = ( MEANS(NLAP,1)   MEANS(NSY,1) ) /
     1          SQRT( ( MEANS(NLAP,2)**2 + MEANS(NSY,2)**2 ) / N3 )

*       Compare t value above with t(0.025) for N3 observers:

        IF( ABS(T).GT.TTEST(N3)) THEN
          WRITE (6,550) T
  550     FORMAT(25X,'Test between NSY     and Mod.N/M images:      ',
     1    't    '. F8.2, ' ;            Hypothesis REJECTED')
        ELSE
          WRITE (6,551) T
  551     FORMAT(25X,'Test between NSY     and Mod.N/M images:      ',
     1    't    ', F8.2, ' ;              CANNOT REJECT')
        END IF

*       Test between Tomita/Tsuji and Modified Tomita/Tsuji algorithms
*       --------------------------------------------------------------
```

```
      T   ( MEANS(TLAP,1)   MEANS(TOM,1) ) /
     1          SQRT( ( MEANS(TLAP,2)**2 + MEANS(TOM,2)**2 ) / N3 )

*     Compare t value above with t(0.025) for N3 observers:

      IF( ABS(T).GT.TTEST(N3)) THEN
        WRITE (6,560) T
  560   FORMAT(25X,'Test between T/T     and Mod.T/T images:      ',
     1    't    ', F8.2, ' ;            Hypothesis REJECTED')
      ELSE
        WRITE (6,561) T
  561   FORMAT(25X,'Test between T/T     and Mod.T/T images:      ',
     1    't  = ', F8.2, ' ;              CANNOT REJECT')
      END IF

*     Test between Nagao/Mats. and Modified Nagao/Mats. algorithms
*     ------------------------------------------------------------

      T = ( MEANS(NLAP,1)   MEANS(NAG,1) ) /
     1          SQRT( ( MEANS(NLAP,2)**2 + MEANS(NAG,2)**2 ) / N3 )

*     Compare t value above with t(0.025) for N3 observers:

      IF( ABS(T).GT.TTEST(N3)) THEN
        WRITE (6,570) T
  570   FORMAT(25X,'Test between N/M     and Mod.N/M images:      ',
     1    't    ', F8.2, ' ;            Hypothesis REJECTED')
      ELSE
        WRITE (6,571) T
  571   FORMAT(25X,'Test between N/M     and Mod.N/M images:      ',
     1    't    ', F8.2, ' ;              CANNOT REJECT')
      END IF

*     Test bet'n Modified Tomita/Tsuji and Modified Nagao/Mats. alg.
*     -------------------------------------------------------------

      T   ( MEANS(NLAP,1)   MEANS(TLAP,1) ) /
     1          SQRT( ( MEANS(NLAP,2)**2 + MEANS(TLAP,2)**2 ) / N3 )

*     Compare t value above with t(0.025) for N3 observers:

      IF( ABS(T).GT.TTEST(N3)) THEN
        WRITE (6,580) T
  580   FORMAT(25X,'Test between Mod.T/T and Mod.N/M images:      ',
     1    't  = ', F8.2, ' ;            Hypothesis REJECTED',////////)
      ELSE
        WRITE (6,581) T
  581   FORMAT(25X,'Test between Mod.T/T and Mod.N/M images:      ',
     1    't    ', F8.2, ' ;              CANNOT REJECT',////////)
      END IF


      STOP
      END
```

APPENDIX 10

DATA REDUCTION RESULTS

```
$ RUN DATARED

)))))))))) What is the name of the file with the raw distances data ? : S05SHARP.DAT
```

********** OBSERVERS' RAW DISTANCES **********

OBSERVERS:	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Image A	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Image B	3.4	0.6	0.8	1.2	-0.2	5.2	1.6	0.0	0.4	0.4	-0.2	1.0	1.4	-1.2	0.0
Image C	5.8	0.2	-0.4	0.6	-0.6	4.0	0.2	-0.2	1.0	0.2	-2.4	2.2	1.4	-1.6	1.0
Image D	7.0	-1.4	-1.6	-1.0	-2.6	-0.2	-5.4	-2.4	-0.8	-1.0	-3.2	5.2	-0.6	-4.4	-0.8
Image E	5.8	-1.4	1.2	0.2	-1.6	3.0	-4.4	-0.4	1.4	-0.6	-1.2	3.6	0.8	-0.8	0.8

********** LINEAR CORRELATION FACTORS BETWEEN OBSERVERS **********

OBSERVERS:	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1.0000	-0.5664	-0.3146	-0.2961	-0.7964	0.1518	-0.6240	-0.6231	0.0918	-0.5479	-0.8523	0.8975	0.0122	-0.7294	0.0548
2	-0.5664	1.0000	0.2335	0.7742	0.8930	0.5302	0.9919	0.7004	0.1239	0.9689	0.5308	-0.8367	0.6156	0.4842	0.2217
3	-0.3146	0.2335	1.0000	0.7191	0.4874	0.6130	0.3362	0.7698	0.7641	0.4384	0.7379	-0.4511	0.6408	0.7856	0.5823
4	-0.2961	0.7742	0.7191	1.0000	0.7592	0.8980	0.8113	0.8449	0.6296	0.8866	0.5952	-0.6501	0.9338	0.6326	0.5804
5	-0.7964	0.8930	0.4874	0.7592	1.0000	0.4083	0.9399	0.9827	0.3103	0.9315	0.7840	-0.9812	0.5629	0.8142	0.3830
6	0.1518	0.5302	0.6130	0.8980	0.4083	1.0000	0.5424	0.5772	0.6794	0.6540	0.2393	-0.2530	0.9622	0.3121	0.5978
7	-0.6240	0.9919	0.3362	0.8113	0.9399	0.5424	1.0000	0.7825	0.2066	0.9863	0.6156	-0.8876	0.6438	0.5908	0.2912
8	-0.6231	0.7004	0.7698	0.8449	0.9027	0.5772	0.7825	1.0000	0.6725	0.8408	0.7855	-0.8504	0.7328	0.9376	0.6799
9	0.0918	0.1239	0.7641	0.6296	0.3103	0.6794	0.2066	0.6725	1.0000	0.3589	0.2517	-0.1859	0.7766	0.6113	0.9511
10	-0.5479	0.9689	0.4384	0.8866	0.9315	0.6540	0.9863	0.8408	0.3589	1.0000	0.6065	-0.8558	0.7573	0.6354	0.4281
11	-0.8523	0.5308	0.7379	0.5952	0.7840	0.2393	0.6156	0.7855	0.2517	0.6065	1.0000	-0.8427	0.3105	0.8540	0.1581
12	0.8975	-0.8367	-0.4511	-0.6501	-0.9812	-0.2530	-0.8876	-0.8504	-0.1859	-0.8558	-0.8427	1.0000	-0.4060	-0.8165	-0.2486
13	0.0122	0.6156	0.6408	0.9338	0.5629	0.9622	0.6430	0.7328	0.7766	0.7573	0.3105	-0.4060	1.0000	0.4872	0.7537
14	-0.7294	0.4842	0.7856	0.6326	0.8142	0.3121	0.5908	0.9376	0.6113	0.6354	0.8540	-0.8165	0.4872	1.0000	0.5967
15	0.0548	0.2217	0.5823	0.5884	0.3830	0.5978	0.2912	0.6799	0.9511	0.4281	0.1501	-0.2486	0.7537	0.5967	1.0000

********** CORRELATION STATISTICSS **********

(Average correlation with other observers and standard deviation)

OBSERVERS:	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
AVG	-0.2959	0.4047	0.4530	0.5799	0.4570	0.4938	0.4448	0.5530	0.4459	0.5064	0.3404	-0.5263	0.5560	0.4426	0.4302
SDEV	0.4877	0.5430	0.3929	0.4657	0.6111	0.3111	0.5681	0.5568	0.3260	0.5542	0.5506	0.4932	0.3692	0.5405	0.3144

```
)))))))))) Which data set do you wish to chose as the reference data set
           for the remainder of the calculations ? (enter observer number): 4
```

********** LINEAR REGRESSION PARAMETERS **********

(Correlation with ref., slope, and intercept)

OBSERVERS:	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
CORR	-0.2961	0.7742	0.7191	1.0000	0.7592	0.8980	0.8113	0.8449	0.6296	0.8866	0.5952	-0.6501	0.9338	0.6326	0.5804
SLOPE	-1.0152	0.8939	0.9697	1.0000	1.0152	2.6667	3.0909	1.0606	0.6667	0.6364	1.0152	-1.6515	1.0152	1.3030	0.5152
INTCP	4.6030	-0.5788	-0.1939	0.0000	-1.2030	1.8667	-2.2182	-0.8121	0.2667	-0.3273	-1.6030	2.7303	0.3970	-1.8606	0.0970

********** ADJUSTED DISTANCES **********

OBSERVERS:	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Image A	4.53	0.65	0.20	0.00	1.19	-0.70	0.72	0.77	-0.40	0.51	1.58	1.65	-0.39	1.43
Image B	1.19	1.32	1.03	1.20	0.99	1.25	1.24	0.77	0.20	1.14	1.38	1.05	0.99	0.51
Image C	-1.18	0.87	-0.21	0.60	0.59	0.80	0.78	0.58	1.10	0.83	-0.79	0.32	0.99	0.20
Image D	-2.36	-0.92	-1.45	-1.00	-1.38	-0.77	-1.03	-1.50	-1.60	-1.06	-1.57	-1.50	-0.98	-1.95
Image E	-1.18	-0.92	1.44	0.20	-0.39	0.42	-0.71	0.39	1.70	-0.43	0.40	-0.53	0.40	0.81

********** MEAN ADJUSTED DISTANCES **********

	Mean	Std Dev	LoLimit	HiLimit
Image A	0.7697	1.2819	-3.0761	4.6155
Image B	0.9365	0.4434	-0.3937	2.2667
Image C	0.4826	0.7418	-1.7429	2.7081
Image D	-1.3870	0.4344	-2.6904	-0.0837
Image E	0.1983	0.8834	-2.4519	2.8484

)))))))))) Do you wish to eliminate data sets (due to negative correlation or outside 3 standard deviations limit) ? (Y/

)))))))))) Which observer's data set do you wish to eliminate ? (enter integer 1-15) : 1

)))))))))) Eliminate another one ? (Y/N) : N

********** MEAN ADJUSTED DISTANCES **********

	Mean	Std Dev	LoLimit	HiLimit
Image A	0.5008	0.7757	-1.8263	2.8279
Image B	0.9187	0.4546	-0.4450	2.2825
Image C	0.6013	0.6042	-1.2113	2.4139
Image D	-1.3174	0.3536	-2.3783	-0.2566
Image E	0.2966	0.8271	-2.1845	2.7778

)))))))))) Do you wish to eliminate data sets (due to negative correlation or outside 3 standard deviations limit) ? (Y/

********** SHIFTED MEAN ADJUSTED DISTANCES **********
Shifted to Noisy = 0.0

		Mean	Std Dev
NSY	(Noisy)	0.0000	0.3536
T/T	(Tomita/Tsuji)	1.6141	0.8271
Mod.T/T	(Modified Tomita/Tsuji)	1.9187	0.6042
N/M	(Nagao/Matsuyama)	1.8182	0.7757
Mod.N/M	(Modified Nagao/Matsuyama)	2.2362	0.4546

********** HYPOTHESIS TESTING **********
Null Ho: u1 = u2 ; Alternate H1: u1 # u2 ; Significance level Alpha = 0.05

t(0.025) value for 14 observers is: 2.06

Test between NSY and T/T images:	t = 6.71 ;	Hypothesis REJECTED
Test between NSY and N/M images:	t = 7.98 ;	Hypothesis REJECTED
Test between T/T and N/M images:	t = 0.67 ;	CANNOT REJECT
Test between NSY and Mod.T/T images:	t = 10.25 ;	Hypothesis REJECTED
Test between NSY and Mod.N/M images:	t = 14.53 ;	Hypothesis REJECTED
Test between T/T and Mod.T/T images:	t = 1.11 ;	CANNOT REJECT
Test between N/M and Mod.N/M images:	t = 1.74 ;	CANNOT REJECT
Test between Mod.T/T and Mod.N/M images:	t = 1.57 ;	CANNOT REJECT

VI. REFERENCES

1. H.C. Andrews, A.G. Tescher and R.P. Kruger, "Image Processing by Digital Computer", IEEE Spectrum, 9(7), (1972).

2. F.C. Billingsley, "Applications of Digital Image Processing", Applied Optics, 9(2), 289 (1970).

3. R.C. Gonzalez and P. Wintz, Digital Image Processing, Addison-Wesley, Reading, Mass., 1977.

4. G.A. Baxes, Digital Image Processing: A Practical Primer, Prentice-Hall, Englewood Cliffs, New Jersey, 1984.

5. D.C.C. Wang, A.H. Vagnucci, and C.C. Li, "Digital Image Enhancement: A Survey", Computer Graphics Image Processing, 24, 363 (1983).

6. J.S. Lee, "Digital Image Smoothing and the Sigma Filter", Computer Graphics Image Processing, 24, 255 (1983).

7. W.K. Pratt, Digital Image Processing, John Wiley, New York, 1978.

8. A. Rosenfeld and A.C. Kak, Digital Picture Processing, 2nd Ed., Vol. 1, Academic Press, New York, 1982.

9. R.E. Graham, "Snow Removal: A Noise-Stripping Process for Picture Signals", IRE Trans. Inf. Theor., IT-8, 129 (1962).

10. D.W. Brown, "Digital Computer Analysis and Display of Radio-nuclide Scan", J. Nucl. Med., 7, 740 (1966).

11. A. Lev, S.W. Zucker, and A. Rosenfeld, "Iterative Enhancement of Noisy Images", IEEE Trans. Syst. Man Cybern., SMC-7, 435 (1977).

12. D.C.C. Wang, A.H. Vagnucci, and C.C. Li, "Image Enhancement by Gradient Inverse Weighted Smoothing Scheme", Computer Graphics Image Processing, 15, 167 (1981).

13. H.C. Andrews, "Digital Image Restoration: A Survey", Computer, 7, 36 (1974).

14. A. Rosenfeld, Picture Processing by Computer, Academic Press, New York, 1969.

15. M. Kuwahara et. al., "Processing of RI-Angiocardiographic Images", Digital Processing of Biomedical Images, Plenum, New York, 1976.

16. F. Tomita and S. Tsuji, "Extraction of Multiple Regions by Smoothing in Selected Neighborhoods", IEEE Trans. Syst. Man Cybern., SMC-7, 107 (1977).

17. M. Nagao and T. Matsuyama, "Edge Preserving Smoothing", Computer Graphics Image Processing, 9, 394 (1979).

18. G.L. Anderson and A.N. Netravali, "Image Restoration Based on a Subjective Criterion", IEEE Trans. Syst. Man Cybern., SMC-6, 845 (1976).

19. J.G. Trussell, "A Fast Algorithm for Noise Smoothing Based on a Subjective Criterion", IEEE Trans. Syst. Man Cybern., SMC-7, 677 (1977).

20. J.W. Tukey, Exploratory Data Analysis, Addison-Wesley, Reading, Massachusetts, 1977.

21. W.H. Beyer, CRC Standard Mathematical Tables, 27th Ed., CRC Press Inc., Boca Raton, Florida, 1984.

22. I. Miller and J.E. Freund, Probability and Statistics for Engineers, 2nd Ed., Prentice-Hall, Englewood Cliffs, New Jersey, 1977.

23. J.D. Gaskill, Linear Systems, Fourier Transforms, and Optics, John Wiley, New York, 1978, pp. 212-216.

24. J.S. Lee, "Digital Image Enhancement and Noise Filtering by Use of Local Statistics", IEEE Trans Pattern Anal Mach Intell, PAMI-2(2), 165 (1980).

25. H.C. Andrews, "Monochrome Digital Image Enhancement", Applied Optics, 15(2), 495 (1976).

26. P.G. Engeldrum, Lecture Notes for R.I.T. Course 09-07-402, "Image Microstructure: Lecture Notes", Fall 1983.

27. Ibid., p.70.

28. E.M. Granger, personal communication, "Correlated Noise", Jul. 11, 1985.

29. P.G. Engeldrum, personal communication, "Multiplicative Noise", Aug. 28, 1985.

30. S.K. Mitra and M.P. Ekstrom, Two-Dimensional Digital Signal Processing, Dowden, Hutchingson & Ross, Stroudsburg, Penn., 1978.

31. T.S. Huang, Two-Dimensional Digital Signal I and II, Springer-Verlag, New York, 1981.

32. E.M. Granger, personal communication, "Data Reduction", Jan. 12, 1986.